ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти магістр

на тему: **«Оцінювання параметрів вразливостей інформаційних систем з використанням засобів Big Data аналізу»**

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи та технології
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти магістр
групи 126ІСТ_мд_22
Павленко Є.І.
Керівник: Поночовний Ю.Л.
Рецензент: Брикун О.М.

Полтава – 2023 року

ВСТУП

Актуальність теми. Протягом останніх десятиліть сучасне суспільство стало значно більш залежним від комп'ютерних систем. Банківські операції, управління торгівлею на ринках, автоматизовані військові та державні системи все більше ґрунтуються на комп'ютерних системах. Це призвело до великого ризику використання різних видів атак, які використовують вразливості у програмному та апаратному забезпеченні, особливо для критичних об'єктів.

В сучасному світі ведуться обширні дослідження у сфері безпеки, спричинені вразливістю програмного забезпечення. Незважаючи на існуючі загрози, суспільство не може відмовитися від використання Інтернету та комп'ютерних мереж у сферах фінансів, політики та війни, оскільки вони надають значні можливості. Однак постійне покращення технологій безпеки не може гарантувати абсолютну захищеність комп'ютерних систем.

Вразливості виявляються в основних операційних системах та програмах. З огляду на постійне виявлення нових вразливостей, ключовим є постійний моніторинг безпеки, оперативне встановлення оновлень та використання інструментів для протидії можливим атакам, що використовують ці вразливості.

Наприклад, вразливість у операційній системі може спричинити витік комерційної інформації та значні фінансові втрати. У таких випадках корисно мати можливість передбачити безпеку комп'ютерної системи та її складових. Одним з методів передбачення безпеки є моделювання процесів виявлення та усунення вразливостей на основі статистичних даних, зібраних протягом життєвого циклу програмних засобів.

Наразі існує багато ресурсів в Інтернеті, які надають інформацію про вразливості. Наприклад, база даних NVD дозволяє точно ідентифікувати уразливі програмні продукти та отримати інформацію про способи атак та загрози. База даних CVE є основним постачальником ідентифікаторів вразливостей, які використовуються іншими базами даних, бюлетенями безпеки та іншими ресурсами.

Роботи вчених, таких як А.Ю. Белобородов, А.В. Горбенко, В.С. Харченко та І.В. Котенко, відображають теоретичні основи та практичне застосування досліджень у цій області.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в межах науково-дослідної роботи «Розвиток підприємництва: управлінські, економічні, інноваційна та правові аспекти» відповідно до договору №9 від 15.05.2023 р. між ТОВ «ПАФ Гарант» та Полтавським державним аграрним університетом (розділ «Обгрунтування показників оцінювання гарантоздатності розподілених інформаційних систем»).

Метою кваліфікаційної роботи є отримання кількісних характеристик часу прояву та критичності на основі вибірки вразливостей з відкритих баз.

Завданнями кваліфікаційної роботи є:

– аналіз стандартів з обліку вразливостей у відкритих репозитаріях та систем їх оцінювання,

– проєктування архітектури Apache Spark кластера та вибір методів обробки неструктурованих даних,

– розробка та налаштування скриптів Apache Spark для отримання даних вразливостей та їх статистичного аналізу.

Об'єктом дослідження є процеси інформаційного обміну при формуванні та модифікації записів у відкритих базах даних вразливостей.

Предметом дослідження є моделі формування і параметризації вразливостей на основі вибірок із відкритих баз даних.

Методи дослідження – проведені в роботі дослідження базуються на методах теорії ймовірності та математичної статистики, системного і марковського аналізу, систем масового обслуговування.

Інформаційна база кваліфікаційної роботи складається з наукових статей, міжнародних аналітичних видань і звітів, матеріалів наукових конференцій інтернет-ресурсів, що місять інформацію про вразливості інформаційних систем, їх параметри та класифікатори.

Елементи наукової новизни полягають у розроблені та досліджені методів формування вибірок вразливостей та їх параметризації шляхом підбору закону розподілу та його параметрів за критерієм максимальної правдоподібності.

Практична значущість роботи полягає в можливості повторного застосування та модифікації розробленого програмного коду для кластеру Big-data Apache Spark з метою формування вибірок вразливостей за заданими критеріями відбору. Отримані результати можуть бути корисними для ІТ фахівців при моделюванні спеціалізованих інформаційних управляючих систем.

Апробація результатів дослідження відбувалася шляхом оприлюднення доповідей на наукових конференціях, семінарах.

Публікації. За результатами проведеного дослідження опубліковано тези: «Використання засобів Big Data для аналізу параметрів вразливостей інформаційних систем», Матеріали XII Міжнар. наук. конференції «Інформаційні технології в енергетиці та агропромисловому комплексі», м. Львів, 04-06 жовтня 2023 р.; «Аналіз заходів захисту інформаційних систем підприємства», Матер. науково-практичної конференції за підсумками виробничої практики здобувачів вищої освіти спеціальності «Інформаційні системи та технології», 17 вересня 2023 р., м. Полтава.

Структура та обсяг кваліфікаційної роботи логічно пов’язані з задачами досліджень. Робота містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг текстової частини дипломної роботи складає 65 сторінок формату А4. Вона містить 20 рисунків і 4 таблиці. В роботі використано 43 науково-технічних джерела.

РОЗДІЛ 1
АНАЛІЗ СТАНДАРТІВ З ОБЛІКУ ВРАЗЛИВОСТЕЙ У ВІДКРИТИХ РЕПОЗИТАРІЯХ ТА СИСТЕМ ЇХ ОЦІНЮВАННЯ

1.1 Аналіз стану розвитку галузі big-data обчислень

Великі дані, або Big Data, – це об'ємні набори інформації, які можуть бути як структуровані, так і неструктуровані [1]. Їх обробляють спеціалізованими автоматизованими інструментами для аналізу, прогнозування та прийняття рішень.

Термін «великі дані» був запропонований редактором журналу Nature, Кліффордом Лінчем, у спецвипуску 2008 року. Він вказав на вибуховий ріст обсягів інформації у світі. Лінч визначив великі дані як будь-які неоднорідні набори даних, що перевищують 150 Гб на добу, проте чіткого критерію ще не встановлено.

До 2011 року аналіз великих даних проводили в рамках наукових та статистичних досліджень. Але починаючи з 2012 року обсяги даних зростали надзвичайно, і з'явилася потреба в їхньому систематичному аналізі та практичному використанні.

Починаючи з 2014 року, провідні освітні заклади світу почали звертати увагу на великі дані, викладаючи їх студентам напряму прикладної інженерії та IT. Пізніше до збору та аналізу даних долучилися IT-корпорації, такі як Microsoft, IBM, Oracle, EMC, а потім і Google, Apple, Facebook та Amazon [2]. Сьогодні великі компанії у всіх галузях, а також урядові органи активно використовують великі дані.

Великі дані є ключовими для аналізу всіх важливих факторів та для прийняття обґрунтованих рішень. За допомогою Big Data розробляють моделі-симуляції для тестування рішень, ідей, продуктів. Основні джерела великих даних включають [3, 4]:

– інтернет речей (IoT) та пристрої, які до нього підключені;

– соціальні мережі, блоги, медіа;

– дані компаній: транзакції, замовлення товарів та послуг, поїздки таксі та каршерингом, профілі клієнтів;

– показники приладів: метеостанції, датчики якості повітря та водойм, дані з супутників;

– статистика міст і країн: дані про переміщення, народження та смертність;

– медичні дані: аналізи, хвороби, діагностичні знімки.

Сучасні обчислювальні системи забезпечують швидкий доступ до великих об'ємів даних. Для їх зберігання використовують спеціалізовані дата-центри з потужними серверами. Крім традиційних фізичних серверів, застосовують хмарні сховища [5], «озера даних» (data lake – сховища великого обсягу неструктурованих даних з одного джерела) [6] та Hadoop – фреймворк для розробки та виконання програм розподілених обчислень [7]. Для роботи з Big Data використовують передові методи інтеграції, управління та підготовки даних для аналізу.

Хмарні технології обчислення є економічними та практичними для організацій, корпорацій, фірм та наукових потреб (рис. 1.1). Вони не потребують великих обчислювальних ресурсів з боку користувача, але вимагають надійного доступу до Інтернету.

Рисунок 1.1 – Структура хмарних обчислень

Серед подібних концепцій обчислювань також можна відзначити [8, 9]:

1. Клієнт-серверна модель – в загальному розумінні це будь-які розподілені програми, які постачаються від сервера до клієнта.

2. Мейнфрейми – це потужні комп'ютери, призначені для обробки великих обсягів завдань.

3. Рівний до рівного (Peer-to-peer) – це розподілена архітектура без централізації.

4. Грід-обчислення – це форма розподіленого та паралельного обчислення, де два або більше віртуальних або супер-комп'ютери об'єднуються в мережу для обчислення складних завдань.

Компанія Meta Group визначила основні характеристики великих даних [13], а саме:

– обсяг даних (Volume) – починаючи від 150 Гб на добу;

– швидкість накопичення та обробки даних (Velocity) – великі дані оновлюються регулярно, потрібні інтелектуальні технології для їх обробки в режимі онлайн;

– різноманітність типів даних (Variety) – дані можуть бути структурованими, неструктурованими або частково структурованими.

Сучасні дослідники додатково визначають три нові ознаки [13, 14]:

– достовірність (Veracity) – якість самого набору даних та результатів його аналізу;

– мінливість (Variability) – піки та спади в потоках даних під впливом сезонних або соціальних явищ. Чим більша мінливість, тим складніше аналізувати дані;

– цінність (Value) – можливість визначити значущість або цінність інформації. Це може бути проста або складна інформація для аналізу, наприклад, від постів у соцмережах до банківських транзакцій.

Динаміка збільшення обсягів і різноманіття даних вимагає постійного вдосконалення аналітичних підходів та інструментів для їх оптимального використання.

1.2 Визначення та стандарти галузі опису вразливостей

Наразі, існують розбіжності у визначенні терміну «вразливість» на рівні стандартів (табл. 1.1). Одні стандарти розглядають вразливість лише як програмний дефект. Однак це погляд не враховує можливість існування програмних дефектів, що ніколи не перетворюються на вразливості (чи то через відсутність доступу для зловмисників, чи через виявлення та усунення під час тестування, коли інформація про них не залишається у репозитаріях вразливостей). Також існують вразливості, що не можна зв'язати з програмними дефектами, такі як слабкі паролі та помилки конфігурації системи. Різницю між вразливостями та програмними дефектами можна побачити на рис. 1.2.

Таблиця 1.1 – Стандартні визначення вразливостей інформаційних систем

Стандарт	Вразливість (vulnerability)	Слабкий місце (weakness)
X.800, X.1500, X.1521	Будь-яке слабке місце, яке може бути використане для порушення системи або інформації, яка в ній міститься	Недолік або дефект, який, хоча і не визнається сам по собі в якості вразливості, але міг би в якийсь момент стати вразливістю або міг би сприяти появі інших вразливостей
X.1520	Будь-який дефект програмного забезпечення, який може бути використаний для порушення цілісності системи або інформації, що міститься в цій системі	
X.1524	Будь-яке слабке місце в програмному забезпеченні, яке може бути використане для порушення системи або інформації, що міститься в ній	Дефект або вада в коді, проєктуванні, архітектурі або розгортанні програмного забезпечення, здатний в певний момент стати вразливістю або сприяти до виникнення інших вразливостей

З 1999 року компанією MITRE Corporation були впроваджені стандарти та засоби ідентифікації та обліку вразливостей, атак, конфігурацій та інших аспектів інформаційної безпеки, що є незалежними від різних виробників [15]. Стосовно до вразливостей, це стандарти CVE (Common Vulnerabilities and Exposures) і CVSS (Common Vulnerability Scoring System).

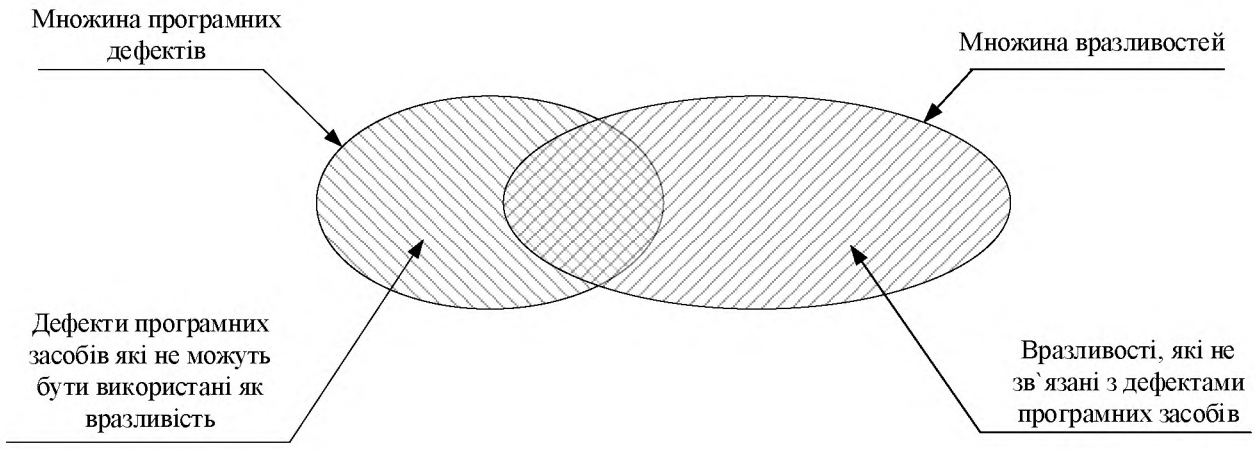

Рисунок 1.2 – Множини вразливостей та програмних дефектів

Зараз існують різноманітні інструменти, які сприяють виявленню вразливостей у системі. Хоча ці інструменти дозволяють аудиторам отримати досить повний огляд можливих системних вразливостей, вони не можуть замінити роль людини у їх оцінці.

Для забезпечення безпеки та недоторканності системи важливо постійно відстежувати її стан: встановлювати оновлення та використовувати інструменти, що сприяють запобіганню можливим атакам. Вразливості можуть бути виявлені в різних операційних системах, таких як Microsoft Windows, Mac OS, різні варіанти UNIX (включаючи GNU/Linux) і OpenVMS [16]. Оскільки нові вразливості постійно з'являються, головний спосіб зменшити ймовірність їх експлуатації – це постійно бути на сторожі та користуватися оновленими версіями програмного забезпечення.

Обмін інформацією про кібербезпеку (CYBEX) є ключовим фактором забезпечення впевненості та безпеки при використанні інформаційно-комунікаційних технологій (ІКТ) [17]. CYBEX уявляє умови обміну і створює ефективну екосистему кібербезпеки, в якій використовуються знання, накопичені звітами, перевірками та практичним досвідом для створення та розвитку інформації про слабкі місця та вразливості. Ці дані разом із звітами про стан системи використовуються для підвищення рівня безпеки.

У онтології CYBEX визначені терміни, такі як операції кібербезпеки, об'єкт кібербезпеки та оперативна інформація про кібербезпеку. Ця онтологія розкриває методи забезпечення кібербезпеки, описані в розділі про CYBEX. Модель охоплює загальний спектр операцій кібербезпеки, включаючи типи, властивості та зв'язки (рис. 1.3).

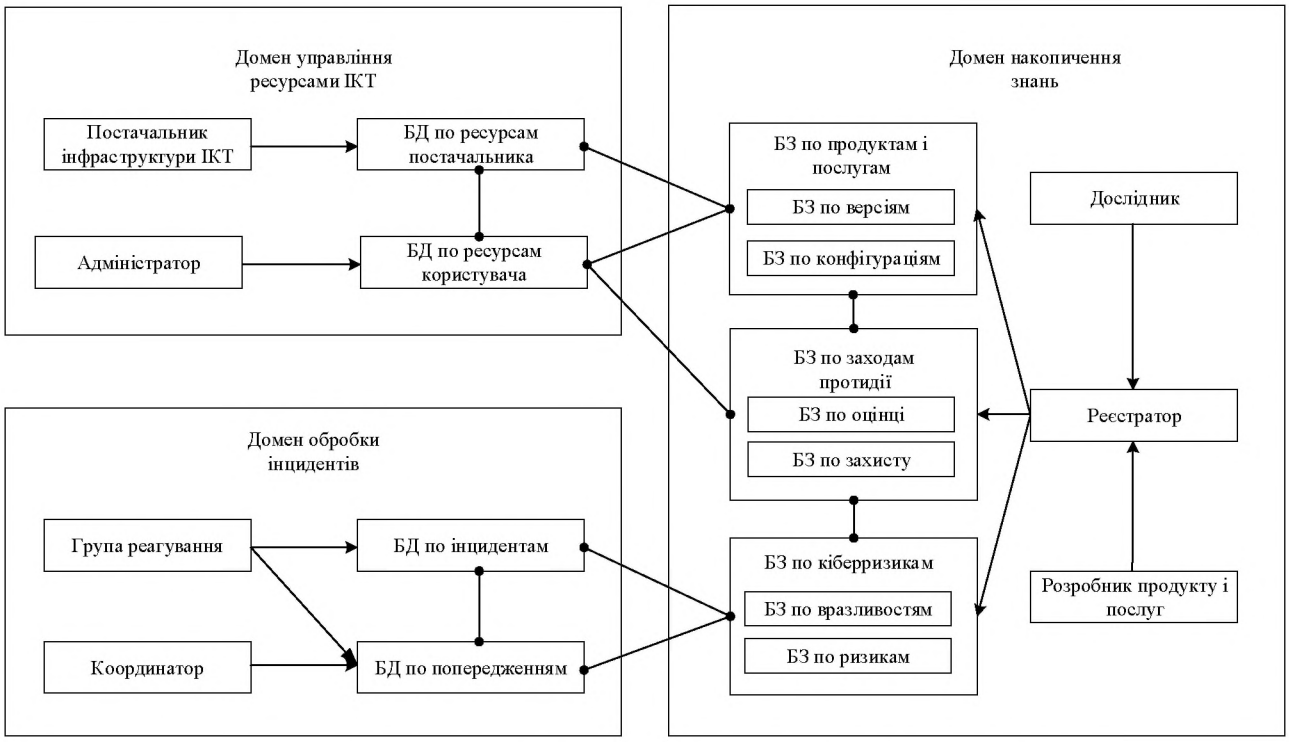

Рисунок 1.3 –Домени та елементи онтології CYBEX

У даній онтології використано модель, що визначає домени операцій кібербезпеки. Ці домени подальше застосовуються для ідентифікації необхідних об'єктів кібербезпеки, що підтримують операції в кожному домені. Наведено детальний опис онтології, де розглянуто, як можуть використовуватися методи CYBEX для забезпечення цієї онтології. Операції кібербезпеки складаються, в основному, з трьох доменів: обробки інцидентів, управління ресурсами ІКТ та накопичення знань.

Домен обробки інцидентів включає виявлення та реагування на інциденти кібербезпеки через моніторинг інцидентів, комп'ютерних подій, які породжують інциденти, та виявленої поведінки під час атак, виявлених у інцидентах. Наприклад, цей домен виявляє аномалії у сповіщеннях від детекторів і об'єднує

детальні дані шляхом збору різних журналів. Часом він генерує сповіщення та надає інструкції, наприклад, ранні попередження про можливі загрози для організацій-користувачів.

Домен управління ресурсами ІКТ включає операції кібербезпеки, що проводяться в межах кожної організації-користувача, такі як встановлення та налаштування ресурсів ІКТ організації, а також їх управління. Цей домен охоплює як операції з запобігання інцидентам, так і операції з контролю шкоди, що здійснюються в кожній організації.

Домен накопичення знань містить інформацію про кібербезпеку. В ньому створюються та накопичуються багаторазово використовувані знання організацій.

З урахуванням вищеописаних доменів операцій можна визначити функціональні об'єкти кібербезпеки, необхідні для виконання операцій кібербезпеки в кожному домені.

У домені обробки інцидентів існують два об'єкти, що відіграють його роль: група реагування та координатор. Група реагування здійснює моніторинг та аналіз різних видів інцидентів, таких як несанкціонований доступ, атаки DDoS і фішинг, а також збір інформації про інциденти. На основі цієї інформації група реагування приймає заходи, наприклад, вносить адреси фішингових сайтів до чорних списків. Координатор координує дії з іншими об'єктами і вирішує проблеми можливих загроз, використовуючи відомості про інциденти.

У домені управління ресурсами ІКТ працюють два об'єкти операцій: адміністратор і постачальник інфраструктури ІКТ. Адміністратор здійснює адміністративне управління системою своєї організації і володіє інформацією про власні ресурси ІКТ. Типовим прикладом адміністратора в межах кожної організації є адміністратор інформаційних технологій. Постачальник інфраструктури ІКТ надає для кожної організації інфраструктуру ІКТ, що включає мережеві зв'язки, хмарні обчислення (наприклад, програмне забезпечення як послуга (SaaS), платформа як послуга (PaaS) і інфраструктура як послуга (IaaS)), а також послуги ідентифікації. Типовими представниками є постачальник послуг Інтернету (ISP) і постачальник застосунків (ASP).

У сфері накопичення знань існують три основні об'єкти діяльності: дослідник, розробник продукту і послуг та реєстратор. Дослідник займається аналізом інформації про кібербезпеку, використовуючи та накопичуючи знання. Розробник продукту і послуг має у себе інформацію про продукти та послуги, такі як назви, версії, їх вразливості, коригування та конфігурації. Ці об'єкти можуть бути представленими як постачальники програмного забезпечення, постачальники послуг (ASP) або індивідуальні розробники програм.

Реєстратори відповідають за класифікацію та систематизацію знань про кібербезпеку, наданих дослідниками, розробниками та постачальниками, для подальшого використання будь-якою іншою організацією.

Враховуючи вищеописані об'єкти та діяльність відповідних доменів, представлені більш детальні дані про оперативну інформацію з кібербезпеки, яка забезпечується функціональними об'єктами в кожному домені.

У сфері обробки інцидентів існують дві бази даних: база даних інцидентів і база даних попереджень. База даних інцидентів містить інформацію про інциденти, представлену групою реагування. Тут знаходяться три типи записів: подія, інцидент і атака. Інформація про подію охоплює комп'ютерні події, такі як вхід привілейованих користувачів. Також відображається інформація про пакети, файли та транзакції, що стосуються інциденту. Більшість записів зазвичай генеруються автоматично. Інформація про інцидент включає події, які можуть бути інцидентами і формується з декількох записів про подію і ручними чи автоматичними гіпотезами. Інформація про атаку ґрунтується на аналізі інцидентів і містить точну дату і час атаки, а також їх послідовність.

База даних попереджень включає в себе інформацію про заходи з кібербезпеки, надані групою реагування та координатором. Попередження ґрунтуються на базі даних інцидентів та знаннях про кіберризики.

У сфері управління ресурсами ІКТ знаходяться дві бази даних: база даних ресурсів користувача і база даних ресурсів постачальника. База даних ресурсів користувача містить інформацію про ресурси в межах конкретної організації, таку як перелік програмного та апаратного забезпечення, їх конфігурації, статус

використання, політику безпеки та інші аспекти. Цю інформацію надає адміністратор.

База даних ресурсів постачальника містить інформацію про зовнішні ресурси, що використовуються за межами організації, такі як хмарні послуги та інші зовнішні ресурси, а також інформацію про зовнішні мережі. Інформація про зовнішні ресурси включає інформацію про їх використання кожною організацією, наприклад, статус та список хмарних послуг. Інформація про зовнішню мережу включає топологію, маршрутизацію, політику доступу, статус трафіку та рівні безпеки. ця інформація надається постачальником ІКТ.

Метод ідентифікації відомих уразливостей та незахищеностей (CVE) – це система, яка спрощує обмін інформацією про вразливості безпеки шляхом призначення загальних ідентифікаторів проблем [18]. Основна мета CVE полягає у сприянні обміну даними між різними інструментами управління вразливостями (засобами, сховищами та сервісами) за допомогою цього «загального списку». Метод спрямований на поєднання баз даних вразливостей та інших ресурсів, а також спрощення порівняння інструментів та послуг безпеки. CVE фактично не містить інформації про ризики, дії, виправлення чи докладну технічну інформацію, а лише надає номер стандартного ідентифікатора з коротким описом та посиланням на відповідні повідомлення та інструкції з вразливостей. Його ціль – вмістити повний обсяг інформації про всі відомі вразливості й незахищеності, у той час як основний акцент зроблено на ідентифікацію вразливостей та незахищеностей, виявлених за допомогою інструментів безпеки, а також на виявлення нових загальних проблем і їх подальше вирішення.

Система оцінки відомих вразливостей (CVSS) надає структуру для подання інформації про характеристики і вплив вразливостей ІКТ [19]. CVSS складається з трьох груп – базової, тимчасової та середовища. Кожній з них присвоюється балова оцінка від 0 до 10 та вектор – стисле текстове представлення значень, використаних для отримання оцінки. Базова група відображає внутрішні якості вразливості, тимчасова – змінні характеристики з часом, а група середовища – зовнішні фактори для користувача. CVSS дозволяє адміністраторам ІКТ, постачальникам бюлетенів

з описом вразливостей, розробникам засобів безпеки, розробникам додатків та дослідникам використовувати спільну мову для оцінки вразливостей ІКТ.

Список загальновідомих слабких місць (CWE) є результатом визначення та обміну уніфікованими даними про слабкі місця в програмному забезпеченні [20]. CWE дозволяє більш ефективно обговорювати, описувати, вибирати та використовувати інструментальні засоби та послуги безпеки для виявлення слабких місць у вихідному коді та операційних системах. Використання стандартної термінології дозволяє постачальникам послуг інформувати користувачів про конкретні слабкі місця та пропонувати шляхи їх усунення, а також дозволяє покупцям програмного забезпечення порівнювати схожі продукти, пропоновані різними розробниками.

Перелік загальновідомих платформ (CPE) є стандартним способом ідентифікації та опису програмних систем і апаратних пристроїв, які наявні у комп'ютерному середовищі підприємства [21]. Ця система забезпечує: точне іменування з логічною структурою імен CPE в правильному форматі, процедури для порівняння цих ідентифікаторів та їх зв'язування, а також словник, що визначає концепцію ідентифікаторів і надає високорівневі правила для кураторів словника.

Перелік загальновідомих конфігурацій (CCE) створює унікальні ідентифікатори для завдань, пов'язаних із зміною системи, для швидкого і точного зіставлення даних конфігурації з різних джерел інформації та інструментів [22]. Наприклад, ідентифікатори CCE можна використовувати для зв'язування результатів перевірок, проведених інструментами оцінки конфігурації, з інструкціями у документах про передовий досвід у цій області.

1.3 Система оцінювання вразливостей CVSS

Необхідно визначити пріоритетність виправлення вразливостей, розташовуючи на першому місці ті, що представляють найбільшу загрозу. Проте, через велику кількість вразливостей та їх оцінку за різними шкалами [9, 13, 24], складно об'єднати ці дані для загального аналізу. Загальна система оцінки

вразливостей (CVSS) є відкритою схемою, призначеною для вирішення цієї проблеми.

CVSS складається з трьох основних метрик: базової, тимчасової та контекстної. Кожна з них включає набір метрик, як зображено на рис. 1.4.

Рисунок 1.4 – Групи метрик CVSS

Групи метрик CVSS пояснюються так:

1. Базові метрики відображають основні характеристики вразливостей, які залишаються постійними з часом і незалежні від середовища.

2. Часові метрики відображають характеристики вразливостей, що можуть змінюватися з часом, але залишаються незалежними від середовища.

3. Метрики середовища відображають характеристики вразливостей, що залежать від конкретного середовища.

Базові метрики CVSS призначені для опису основних аспектів вразливостей, надаючи користувачам чітке та легко зрозуміле уявлення про них. Використовуючи часові та контекстні метрики, користувачі можуть отримати детальну інформацію про вразливості, враховуючи їхнє специфічне середовище. Це полегшує прийняття обґрунтованих рішень щодо мінімізації ризиків, пов'язаних з вразливостями.

Існують інші системи оцінки вразливостей, створені різними організаціями, які мають свої переваги. Наприклад, CERT/CC використовує значення оцінок від 0 до 180, враховуючи ризики Інтернет-інфраструктури та необхідність передумов для експлуатації вразливостей [25]. SANS оцінює вразливості залежно від їхньої конфігурації та ролі в мережі [26]. Система оцінки від Microsoft враховує

складність експлуатації та загальний вплив від експлуатації вразливостей [9]. Ці системи корисні, але вони мають певний недолік – вони вважають наслідки експлуатації вразливостей однаковими для приватних осіб і компаній.

Систему CVSS можна також описати у контексті її обмежень:

– вона не є системою оцінки загроз, як, наприклад, системи, що використовуються Міністерством безпеки США та центром Sans Internet Storm Center, які надають систему попереджень про небезпеки для критично важливих IT-мереж [27];

– вона не є базою даних вразливостей, як National Vulnerability Database (NVD), Open Source Vulnerability Database (OSVDB) або Bugtraq, що являють собою каталоги відомих вразливостей з додатковою інформацією [28];

– вона не є системою ідентифікації вразливостей, такою як стандарт Common Vulnerabilities and Exposures (CVE) або словник вразливостей Common Weakness Enumeration (CWE), які однозначно ідентифікують вразливості відповідно до їхнього місця виявлення [29].

Коли базові метрики визначені, використовуючи базову формулу, обчислюється оцінка від 0 до 10, і створюється вектор для точного відображення отриманої оцінки (рис. 1.5). Вектор – це текстовий рядок, що містить значення, пов'язані з кожною метрикою, і використовується для пояснення отриманої оцінки. Тому важливо публікувати вектор разом із оцінкою.

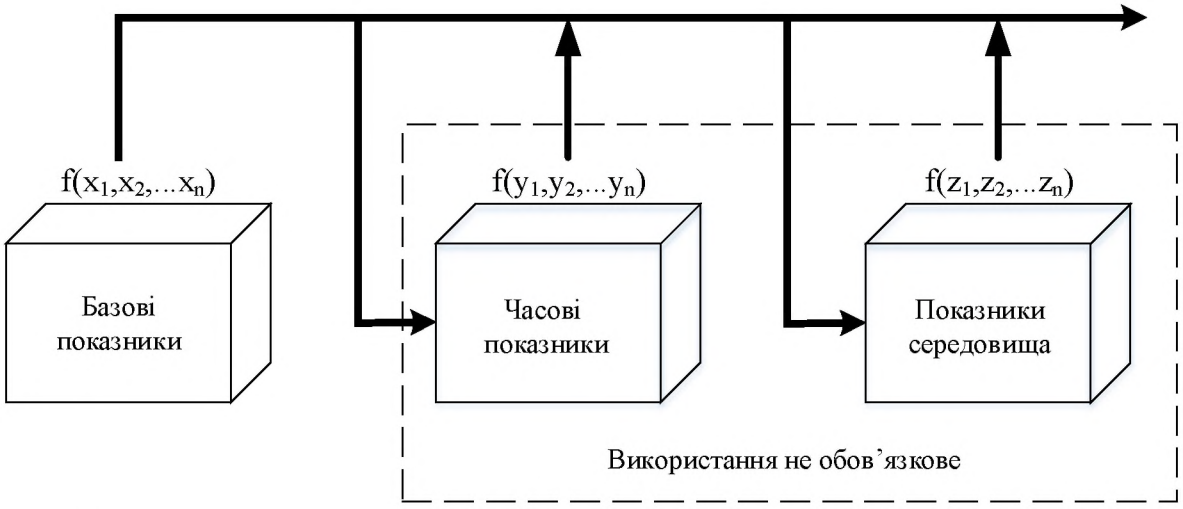

Рисунок 1.5 – Послідовність розрахунку та обов'язковість оцінок CVSS

Для обчислення часової метрики застосовується формула, яка комбінує часові показники з базовими, утворюючи часову оцінку від 0 до 10. Крім того, для розрахунку контекстної оцінки застосовується контекстна формула, яка поєднує контекстні метрики з тимчасовою оцінкою, утворюючи контекстну оцінку також у діапазоні від 0 до 10. Зазвичай аналітики бюлетенів вразливостей, виробники продуктів у сфері інформаційної безпеки або розробники додатків визначають базові та часові метрики, оскільки вони мають більш повну інформацію про характеристики вразливостей, ніж звичайні користувачі. Проте, контекстні метрики визначаються саме користувачами, оскільки вони можуть оцінити наслідки вразливості у власному середовищі.

CVSS розробляється Форумом інцидентів та команд реагування на загрози та безпеку (FIRST) [30]. Проте, варто зауважити, що це вільний та відкритий стандарт. Жодна організація не має прав на CVSS, і членство в FIRST не зобов'язує використовувати або впроваджувати CVSS. Єдиною вимогою від FIRST до організацій, які публікують оцінки, є узгодженість оцінок із правилами та одночасне публікування оцінок і вектору, щоб користувачі могли зрозуміти, як саме була отримана конкретна оцінка.

1.4 Репозиторії вразливостей CVE та NVD

CVE (Common Vulnerabilities and Exposures) – це словник публічно оприлюднених вразливостей і ризиків кібербезпеки [31]. Це зусилля спільноти, спрямоване на стандартизацію ідентифікації вразливостей у різних системах і постачальниках. CVE призначає унікальний ідентифікатор кожній вразливості або впливу, що допомагає дослідникам і фахівцям із безпеки спілкуватися та відстежувати їх.

CVE вперше було створено в 1999 році корпорацією MITRE, некомерційною організацією, яка керує проєктами досліджень і розробок для уряду США [32]. З тих пір він став широко визнаним і прийнятим стандартом для відстеження вразливостей і керування ними.

Ідентифікатори CVE мають певний формат, який включає рік виявлення вразливості та унікальний номер. Наприклад, CVE-2023-1234 посилатиметься на вразливість, виявлену в 2023 році, з номером ідентифікатора 1234. Однією з переваг використання ідентифікаторів CVE є те, що на них можна легко посилатися в порадах щодо безпеки, базах даних уразливостей та інших ресурсах, пов'язаних із безпекою. Це дає змогу дослідникам і фахівцям із безпеки швидко виявляти конкретні вразливості та повідомляти про них, що може допомогти організаціям визначити пріоритетність своїх зусиль із керування вразливістю.

У розробці CVE крім експертів MITRE брали участь фахівці ISS, Cisco, BindView, Axent, NFR, L-3, CyberSafe, CERT, Carnegie Mellon University, інститут SANS, UC Davis Computer Security Lab, CERIAS і т.д. за підтримки компаній Internet Security Systems, Cisco, Axent, BindView, IBM та інші. У даній роботі використовується переклад стандартів і термінів відповідно до введених в 2012 році рекомендацій МСЕ (ITU-T) серії X-15xx [33].

На сьогодні існує кілька систем класифікацій вразливостей (vulnerabilities), які активно використовуються в освітніх і технологічних процесах. Однією з найвідоміших і найбільш повною є CVE (Common Vulnerabilities and Exposures), яка курується компанією NCSD (National Cyber Security Division). Повністю CVE розміщена на сервері Національної Бази вразливих США (NVD) або на офіційному сайті.

NVD (National Vulnerability Database) – це фінансована урядом США онлайн-база даних, яка слугує сховищем інформації про загальновідомі вразливості та ризики кібербезпеки. NVD керує Національний інститут стандартів і технологій (NIST), який є підрозділом Міністерства торгівлі США [34].

NVD надає доступну для пошуку базу даних ідентифікаторів CVE, а також інформацію про серйозність кожної вразливості, уражені системи чи програмне забезпечення та кроки для усунення вразливості. NVD також надає посилання на зовнішні ресурси, такі як рекомендації щодо безпеки та виправлення, які можуть допомогти організаціям зменшити ризики, пов'язані з певними вразливими місцями.

Одна з головних переваг NVD полягає в тому, що він надає централізоване джерело інформації про відомі вразливості та ризики, до якого можуть отримати доступ фахівці з безпеки, дослідники та організації будь-якого розміру. Це допомагає гарантувати, що всі зацікавлені сторони мають доступ до тієї самої інформації, що може підвищити ефективність і результативність зусиль з управління вразливістю.

CVE та NVD є відкритими ресурсами, які є у вільному доступі. Кожен може отримати доступ до інформації, яку він надає, що сприяє прозорості та співпраці в спільноті кібербезпеки.

Ідентифікатори CVE використовуються не лише для виявлення вразливостей у програмному забезпеченні та системах, але також можуть використовуватися для виявлення вразливостей у апаратному забезпеченні, вбудованому програмному забезпеченні та інших типах технологій. Це допомагає гарантувати, що всі типи вразливостей відстежуються та керуються послідовним чином.

NVD надає різноманітні параметри пошуку, включаючи можливість пошуку за типом уразливості, рівнем серйозності, ураженим постачальником тощо. Це полегшує пошук інформації про конкретні вразливості та ризики, які можуть мати відношення до конкретної організації чи системи.

І CVE, і NVD постійно оновлюються новою інформацією, коли виявляються та розкриваються вразливості. Це означає, що організаціям важливо бути в курсі найновішої інформації, щоб ефективно керувати зусиллями з управління вразливістю.

Окрім CVE та NVD, існують інші бази даних уразливостей і ресурси, які організації можуть використовувати для керування своїми вразливостями. До них належать комерційні сканери вразливостей, програми винагород за помилки та канали аналізу загроз.

Хоча CVE та NVD є цінними ресурсами, вони не повинні бути єдиним центром зусиль організації з управління вразливістю. Важливо також проводити регулярні оцінки вразливостей, своєчасно виправляти системи та програмне забезпечення, а також застосовувати найкращі методи безпеки, щоб зменшити ризик використання вразливостей.

Нарешті, важливо зазначити, що не всі вразливості розкриваються публічно або їм призначається ідентифікатор CVE. Це означає, що можуть існувати вразливості, які не включені в NVD або інші бази даних уразливостей. Тому важливо проводити комплексну оцінку вразливості та використовувати різні ресурси, щоб гарантувати, що всі вразливості виявлені та керовані ними.

1.5 Життєвий цикл вразливостей

Наразі в стандартах інформаційної безпеки розглядається лише життєвий цикл вразливостей, враховуючи сховища так званої «білої зони». Згідно з Vulnerability Disclosure Framework [34], існує дев'ять етапів цього циклу:

1. Дослідження: в результаті виявляється можливість здійснення атаки через відповідний експлойт.

2. Перевірка: впевнює, що вразливість не є випадковою помилкою, а властивістю системи, яку можна використати.

3. Повідомлення власнику системи: відбувається безпосередньо або через координатора.

4. Оцінка власника системи: для підтвердження висновків дослідника.

5. Підтвердження: результат позитивної оцінки, що сигналізує про подальший контакт із дослідником для майбутніх обговорень.

6. Усунення: розробка рекомендацій або патча для уразливості.

7. Тестування: патчів та рекомендацій для підтвердження їх безпечності для системи.

8. Випуск патча і рекомендацій: їх доступність для користувачів.

9. Зворотній зв'язок і закриття кейса: завершення життєвого циклу вразливості.

Проте, не завжди кожна вразливість пройде всі 9 етапів [35]. Це залежить від дослідника, який може не повторити свої дії, і від виробника, який може вирішити не випускати патч або не повідомляти про це розробників і користувачів. Час між

створенням експлойту та усуненням вразливості (розробкою патча) називають «вікном можливості» (рис. 1.6), коли хакер може використати незакриту вразливість для доступу до системи користувача без його відома [36, 38].

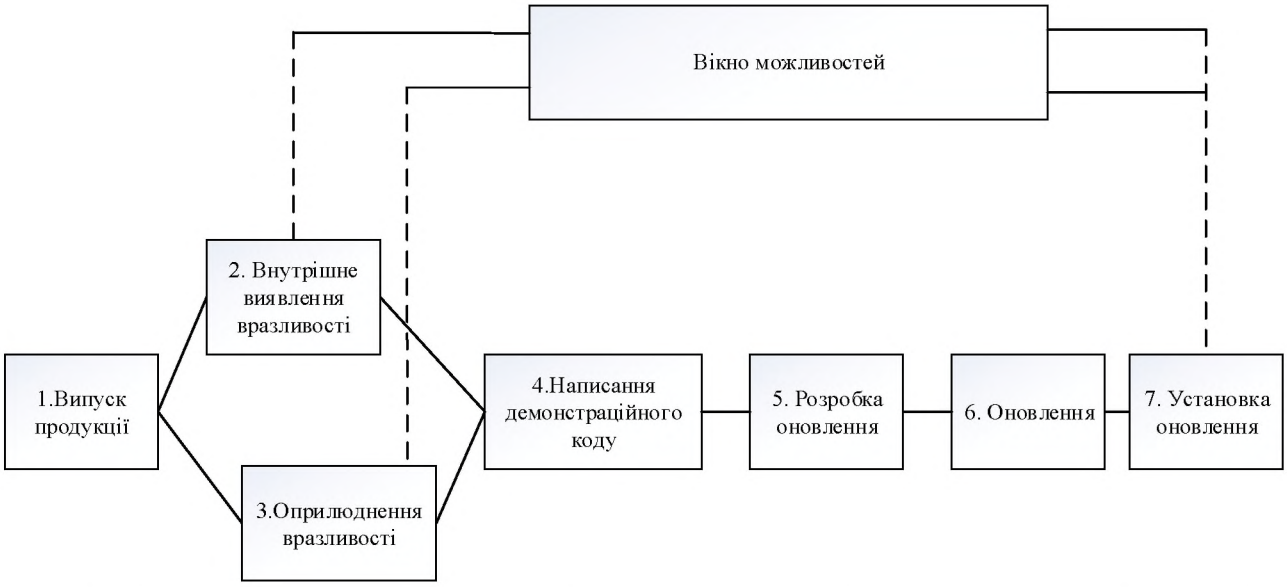

Рисунок 1.6 – Діаграма життєвого циклу програмної уразливості і експлойта

Період, що проймає від виявлення до усунення вразливості, визначають як період ризику, який оцінюється індивідуально для кожної комп'ютерної системи. Ймовірність успішної атаки на вразливість значно зростає з появою експлойтів чи іншого шкідливого програмного забезпечення, що може використовувати цю уразливість. Тому ризик збільшується з часом до випуску виправлення, як це вказано в [39]. Розмір ризику також залежить від наявності додаткових захисних засобів, таких як антивірусне програмне забезпечення, системи виявлення атак чи мережеві екрани.

Важливим питанням залишається вплив публічного розголошення уразливості, такого як публікація у відкритих базах даних CVE. З одного боку, зловмисники можуть використовувати цю інформацію для атаки на комп'ютерні системи та розробки експлойтів. З іншого боку, інформовані користувачі можуть прийняти додаткові заходи для зменшення ризику інформаційних вторгнень, а розробники отримують додаткові дані і спонукання для швидкого випуску виправлень.

Висновки до розділу 1

У першому розділі виконано аналіз сучасного стану галузі обчислень великих даних. У галузі big-data обчислень спостерігається стрімкий розвиток, обумовлений зростанням обсягів даних, що обробляються, та різноманіттям їх джерел. Для ефективного використання великих даних необхідно застосовувати спеціальні технології та методи, які постійно вдосконалюються.

У галузі опису вразливостей існує ряд стандартів, які дозволяють забезпечити уніфікований підхід до їх ідентифікації, оцінки та управління. Найважливішими з них є CVE, CVSS, CWE, CPE та CCE. CVSS є найбільш широко використовуваною системою оцінки вразливостей. Вона дозволяє оцінити вразливості за трьома основними групами метрик: базової, тимчасової та контекстної. Базова група відображає основні характеристики вразливостей, тимчасова – характеристики, що можуть змінюватися з часом, а контекстна – характеристики, що залежать від конкретного середовища. Використовуючи CVSS, можна визначити пріоритетність виправлення вразливостей, розташовуючи на першому місці ті, що представляють найбільшу загрозу.

На основі проведеного аналізу сформульовані загальне завдання, яке полягає у розробці програмного комплексу для автоматизації отримання кількісних характеристик вразливостей (часу прояву та критичності) на основі вибірки даних з відкритих баз. Загальне завдання розділено на три часткові задачі, дві з яких розглянуті у наступних розділах.

РОЗДІЛ 2
ПРОЄКТУВАННЯ АРХІТЕКТУРИ APACHE SPARK КЛАСТЕРА ТА ВИБІР МЕТОДІВ ОБРОБКИ НЕСТРУКТУРОВАНИХ ДАНИХ

2.1 Вибір моделі розгортання кластера для обробки великих даних

Паралельна обробка даних включає виконання одночасно декількох завдань, які утворюють велике завдання. Основна мета полягає в скороченні часу виконання завдання за рахунок розбиття його на кілька менших, що обробляються паралельно. Хоча такий тип обробки може застосовуватися з використанням різних мережевих машин, він частіше реалізується на одній і тій же машині з кількома процесорами або ядрами, згідно із зображенням на рис. 2.1.

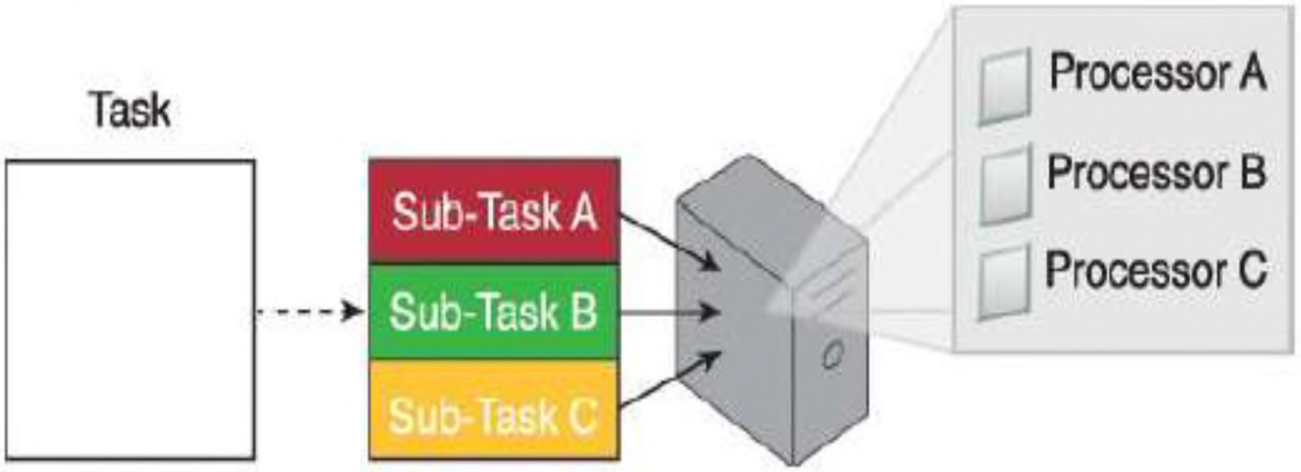

Рисунок 2.1 – Розподіл завдання на три підзадачі при паралельній обробці даних

Розподілена обробка даних тісно пов'язана з паралельною обробкою даних, оскільки в обох випадках використовується загальний принцип «розділяй і володарюй». Але розподілена обробка завжди здійснюється на фізично окремих машинах, які об'єднані в мережу у вигляді кластера. На рис. 2.2 завдання розбивається на три підзавдання, які потім виконуються на трьох різних комп'ютерах, що спільно використовують один фізичний комутатор.

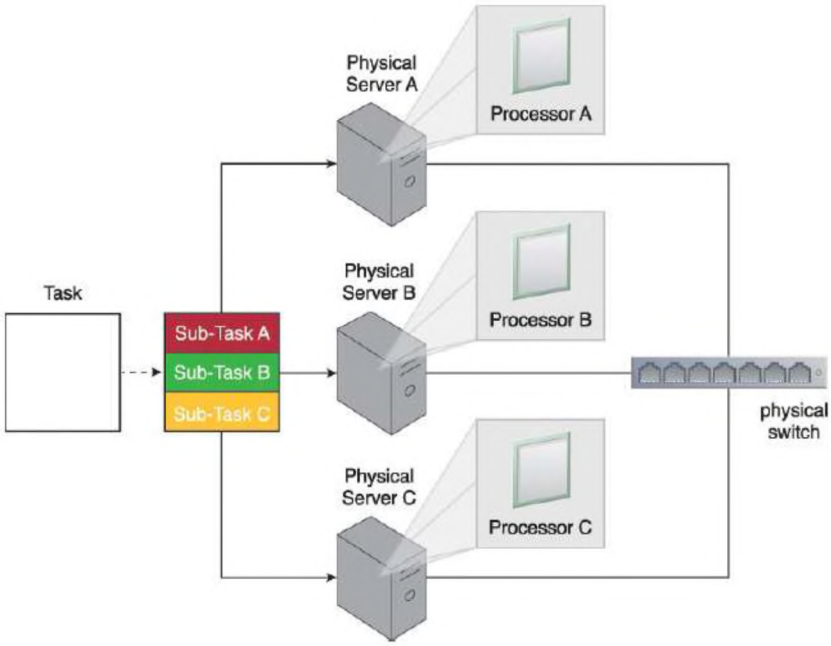

Рисунок 2.2 – Приклад розподіленої обробки даних

Hadoop – це платформа з відкритим вихідним кодом, призначена для масштабного зберігання і обробки даних, яка сумісна зі стандартним апаратним забезпеченням [40]. Платформа Hadoop визнана як важливий стандарт для рішень у сфері великих даних, здатна служити як механізм ETL, так і засіб аналітики для обробки великих обсягів різних типів даних: структурованих, напівструктурованих і неструктурованих. У контексті аналізу, Hadoop використовує інфраструктуру обробки MapReduce. На рис. 2.3 показано деякі особливості платформи Hadoop.

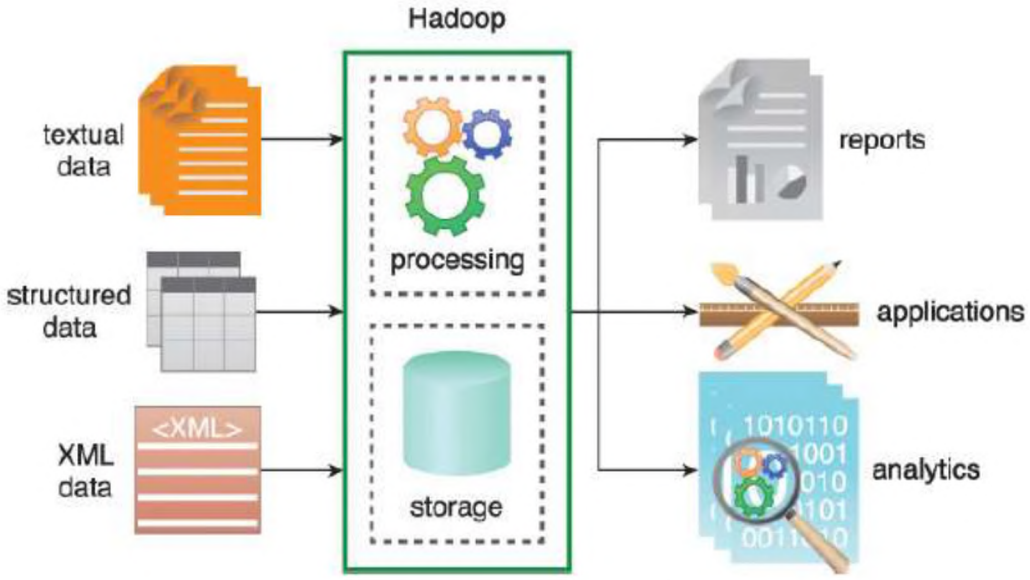

Рисунок 2.3 – Структура та можливості Hadoop

Робочі завдання для обробки великих даних визначаються обсягом та характером даних, які обробляються протягом певного періоду часу. Ці завдання зазвичай розділяються на два основних типи:

1. Пакетна обробка: Вона відома також як автономна обробка, що включає опрацювання даних у великих пакетах, часто призводячи до затримок у відповідях через цей підхід. Цей тип обробки завдань має справу з великими обсягами даних, використовуючи послідовне читання/запис та операції, зокрема, групові запити на читання або запис.

2. Транзакційна обробка: Це тип операцій, який зазвичай включає в себе окремі, відокремлені дії над даними. Він спрямований на більш інтерактивні процеси, які вимагають миттєвих або швидких відповідей у реальному часі.

Пакетна обробка, як показано на рис. 2.4, описує групові операції зчитування/запису, які використовують великі обсяги даних, можуть містити складні зв'язки і надають відповіді з помітною затримкою.

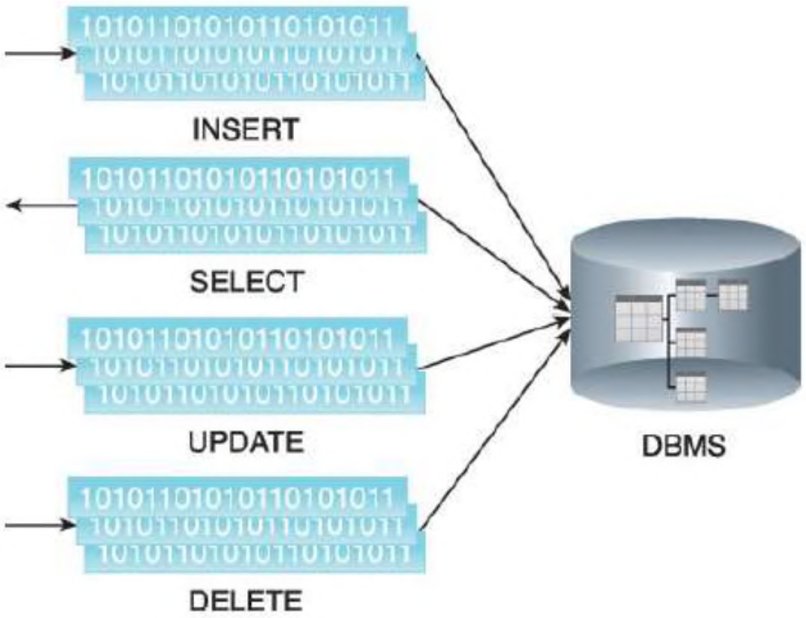

Рисунок 2.4 – Групові операції читання/запису INSERT, SELECT, UPDATE і DELETE для пакетного робочого завдання

Транзакційна обробка, також відома як онлайн-обробка, це метод обробки робочих завдань, що відбувається у реальному часі без затримок у відповідях. Цей

тип завдань включає невеликі обсяги даних і звернення до пам'яті для зчитування та запису.

Транзакційна обробка, яка охоплює такі операційні системи, як OLTP, зазвичай використовує ефективні операції запису. Ці завдання часто використовують операції запису більш ефективно, хоча вони також можуть містити запити на читання. Вони виконують операції зчитування/запису зі зверненням до пам'яті, використовуючи менше з'єднань, ніж завдання бізнес-аналітики та звітів.

Для горизонтально масштабованих рішень у зберіганні даних кластери використовуються як механізм розподіленої обробки даних з лінійною масштабованістю. Це дозволяє розподіляти великі набори даних на менші частини та обробляти їх одночасно. Кластери дозволяють обробляти великі дані як у пакетному, так і у реальному часі (рис. 2.5).

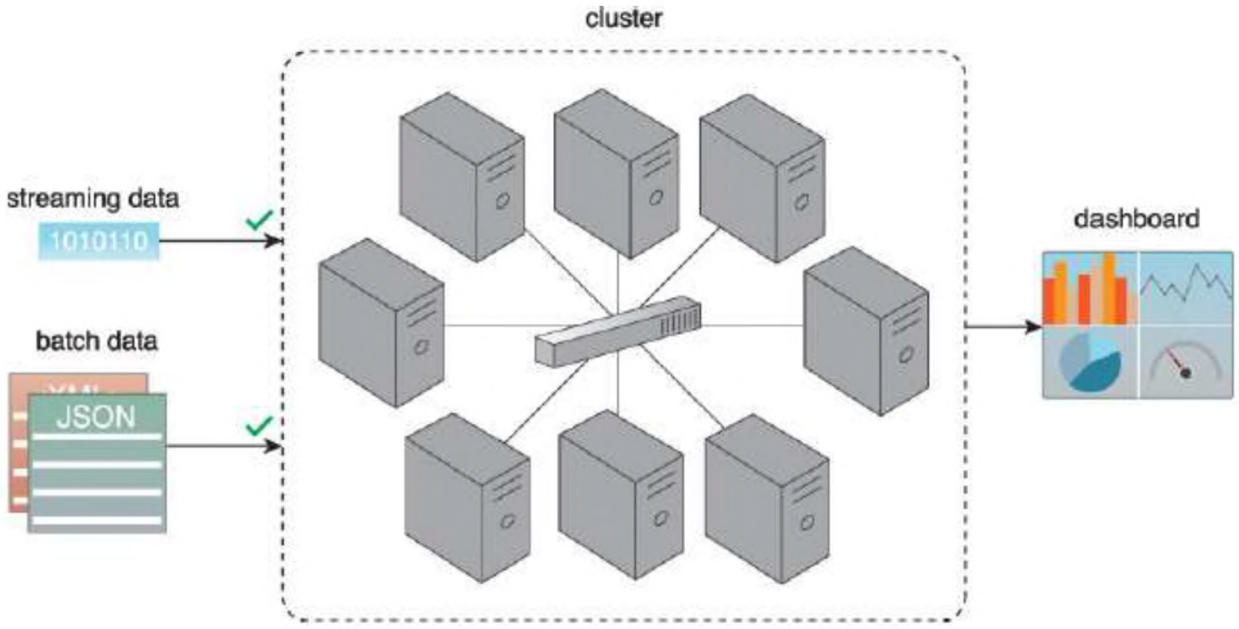

Рисунок 2.5 – Використання кластера для підтримки пакетної обробки великих масивів даних і обробки потокових даних в режимі реального часу

Ці кластери складаються з доступних та масштабованих вузлів, що спільно забезпечують велику обчислювальну потужність. Крім того, вони забезпечують надмірність та відмовостійкість, що дозволяє гнучко виконувати обробку навіть при збоях в мережі чи вузлах. Використання хмарних інфраструктурних сервісів

або готових аналітичних середовищ для цих кластерів стає цікавою перспективою завдяки їхній еластичності та моделям оплати за використання обчислень.

Apache Spark – це платформа паралельної обробки, яка активно використовує пам'ять для поліпшення продуктивності великих аналітичних додатків. Одна з пропозицій Apache Spark в хмарі Azure, відома як Apache Spark в Azure HDInsight [41], дозволяє створювати кластери Spark у співвідношенні з Azure BLOB-об'єктами, Azure Data Lake Storage першого та другого покоління, щоб застосовувати функціонал Spark до існуючих сховищ даних.

Spark має функціонал для розподілених обчислень в пам'яті, що дозволяє завантажувати дані, зберігати їх у кеші пам'яті та декілька разів використовувати [42]. Обчислення в пам'яті працює набагато швидше, ніж у додатках, які оперують дисками, наприклад, у Hadoop з його доступом через розподілену файлову систему HDFS. Spark також інтегрується з мовою програмування Scala, що дає можливість працювати з розподіленими наборами даних, навіть коли вони є локальними колекціями. Це означає, що не потрібно створювати операції порівняння та редукції для обміну даними.

Кластери Apache Spark в HDInsight включають у себе наступні компоненти (рис. 2.6): ядро Spark з Spark SQL, API для потокової обробки Spark, GraphX і MLlib, Anaconda, Apache Livy, записну книжку Jupyter, записну книжку Apache Zeppelin. Ці кластери також мають драйвер ODBC для забезпечення зв'язку з інструментами бізнес-аналітики, такими як Microsoft Power BI [43].

Задачі Spark виконуються як окремі групи процесів у кластері, і вони керуються об'єктом SparkContext, що є частиною основної програми, відомої як програма драйвера. SparkContext може з'єднуватися з різними типами кластерних менеджерів, які розподіляють ресурси між різними програмами. До цих менеджерів відносяться Apache Mesos, Apache Hadoop YARN і Spark. У HDInsight Spark використовує менеджер кластера YARN. Коли Spark під'єднується, він знаходить виконавців на робочих вузлах кластера. Ці виконавці – це процеси, які обирають та зберігають дані для програм. Далі Spark надсилає код програм (зазначений у

файлах JAR або Python, переданих у SparkContext) виконавцям. На завершення, SparkContext надсилає виконавцям підзадачі для виконання.

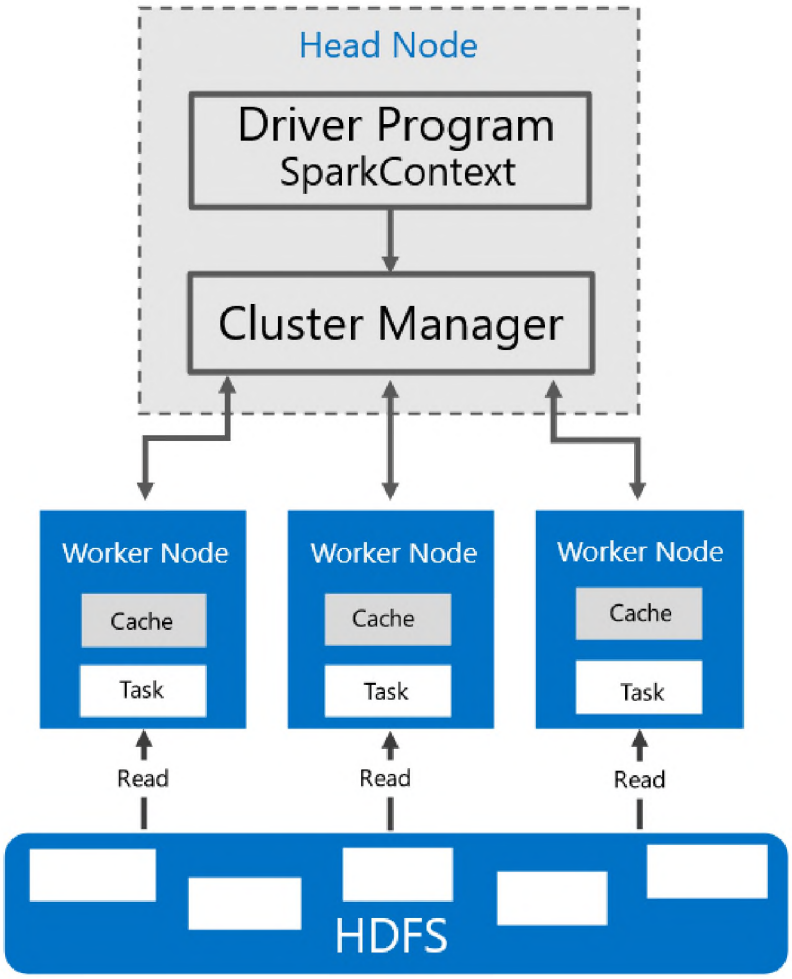

Рисунок 2.6 – Архітектура кластера Spark [27]

SparkContext здійснює основну функцію користувача та виконує різні паралельні операції на робочих вузлах. Після цього SparkContext збирає результати операцій. Робочі вузли обробляють дані з розподіленої файлової системи Hadoop і записують їх. Крім того, робочі вузли кешують перетворені дані як стійкі розподілені набори даних (RDD).

SparkContext підключається до головного вузла Spark і відповідає за перетворення програм в орієнтований граф (DAG) для окремих завдань. Ці завдання виконуються в рамках процесу виконання на робочих вузлах. Кожна програма отримує окремі виконавці, які залишаються активними під час виконання і обробляють завдання в кількох потоках.

2.2 Аналіз математичного апарату оцінювання числових вибірок

Оцінка числових вибірок у програмуванні може бути узгоджена з методами виявлення та усунення програмних дефектів. На сьогоднішній день існує багато моделей надійності програмного забезпечення (ПЗ) та їх модифікацій, які визначають функцію зміни показників надійності. Зазвичай, ці показники включають інтенсивність відмов та кількість виявлених дефектів, обчислені з урахуванням часових рядів тестування та експлуатації ПЗ. Для цих розрахунків використовуються інструментальні системи, такі як SRATS (Software Reliability Assessment Toolon Spreadsheet).

Моделі надійності базуються на припущеннях про процес виявлення дефектів ПЗ. Більшість моделей вважає цей процес випадковим, що відповідає уявленням тестувальників та користувачів. Проте, обґрунтування цих уявлень залишається відкритим питанням, яке потребує окремого вивчення.

У літературі, ймовірнісні моделі зазвичай відомі як «моделі зростання надійності ПЗ» (Software Reliability Growth Model – SRGM). Вони поділяються на дві групи: моделі, що передбачають марковський процес виявлення дефектів, та моделі, що ґрунтуються на неоднорідному пуасонівському процесі (NHPP-моделі).

Марковський процес виявлення дефектів передбачає, що швидкість виявлення дефектів визначається лише кількістю дефектів, які існують у програмі на даний момент часу, не залежно від історії тестування. Якщо ймовірність виявлення дефектів змінюється з часом, це відображає неоднорідний пуасонівський процес. SRATS реалізує NHPP-моделі, які можуть бути опуклими, S-образними або нескінченними, в залежності від швидкості зміни кривої виявлення дефектів.

S-образні моделі вказують, що на ранніх етапах тестування виявлення дефектів менш ефективне через взаємопов'язані помилки та налаштування системи для коректної роботи програми. Тому на початкових етапах зміна кількості дефектів менш динамічна, ніж у середньому етапі тестування.

У деяких моделях припускають, що кількість помилок в програмі є нескінченною. В такому випадку, зв'язок між кумулятивною кількістю виявлених дефектів та часом задається формулою $\mu(t) = \alpha \cdot t^{\beta}$, а інтенсивність відмов – формулою (2.1).

$$\lambda(\mathrm{t}) = \frac{d\mu(t)}{dt} = \alpha \cdot \beta \cdot t^{\beta-1} \tag{2.1}$$

Це припущення призводить до ситуації, коли β=1, λ(t)=const, а μ(t) змінюється за лінійним законом. Якщо β<1, інтенсивність відмов зменшується з часом, а при β>1 – збільшується. Отже, незважаючи на нелогічність припущення про нескінченну кількість дефектів, використання нескінченних моделей, завдяки різноманітності параметра β, дозволяє описати різні випадки зміни інтенсивності відмов і кумулятивних кривих, які зустрічаються на практиці. Це відображення різноманітності реальних сценаріїв стало причиною застосування таких моделей.

2.3 Класифікація баз вразливостей

База даних вразливостей – це платформа, яка призначена для збору, зберігання і розповсюдження інформації про виявлені уразливості, спрямовані на реальні комп'ютерні системи. Зазвичай в таких базах описується виявлена уразливість, міститься оцінка потенційного збитку для комп'ютерних систем і методи захисту від цих вразливостей.

При дослідженні баз даних вразливостей була розроблена модель доступу до них (рис. 2.7). Ця модель показує, що бази даних можуть мати два основних види доступу: відкритий та закритий.

Бази вразливостей із закритим доступом, як правило, використовуються різними сканерами. Ці сканери перевіряють систему на наявність відомих вразливостей та повідомляють користувача. Зазвичай такі сканери потрібно

купувати від виробника, і не завжди вони дають повний доступ до всієї інформації в базі даних. Хоча виробники можуть надавати пробний доступ до бази даних безкоштовно, доступ до неї залишається обмеженим через тимчасовість такої можливості.

Відкриті бази даних цілком безкоштовні і не обмежують отримання повної інформації, але іноді такі бази тимчасово не публікують або обмежують інформацію про вразливості. Це пов'язано з тим, що виробники надають час на створення оновлень або пошук способів захисту користувачів від зловмисників.

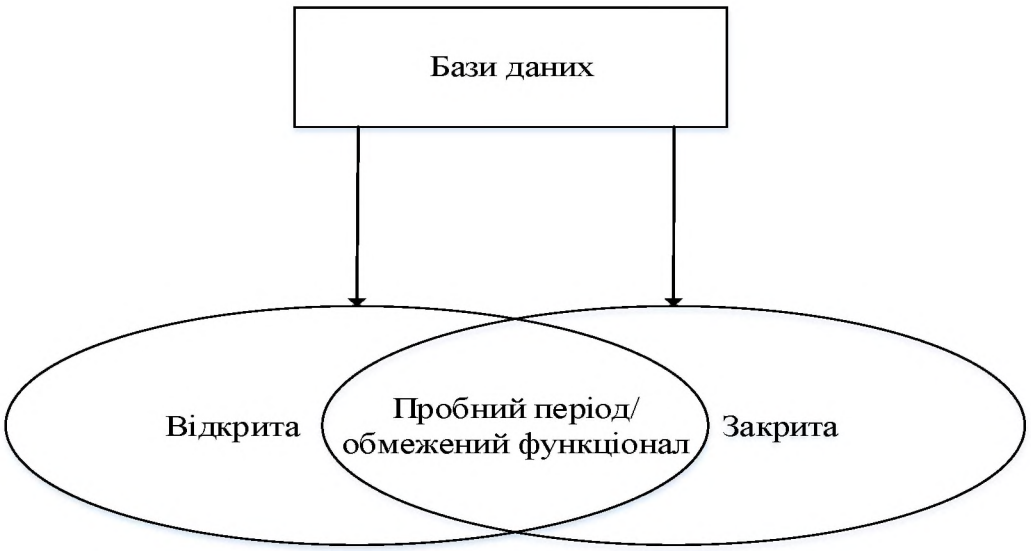

Рисунок 2.7 – Модель доступу до БД вразливостей

Під час дослідження була створена модель для класифікації баз даних за їхнім типом (рис. 2.8). Бази даних можна поділити на два типи:

1. Універсальні – це бази, у яких відсутні основні критерії для відбору вразливостей у свою структуру.

2. Спеціалізовані – вони об'єднані загальною темою, яка характеризує їх вразливості.

Серед спеціалізованих баз даних виділяють два підтипи:

– загальні за виробником – це бази, що містять вразливості лише одного виробника;

– спеціалізовані за продуктом – це бази, які об'єднують вразливості, пов'язані з одним типом продуктів, але від різних виробників.

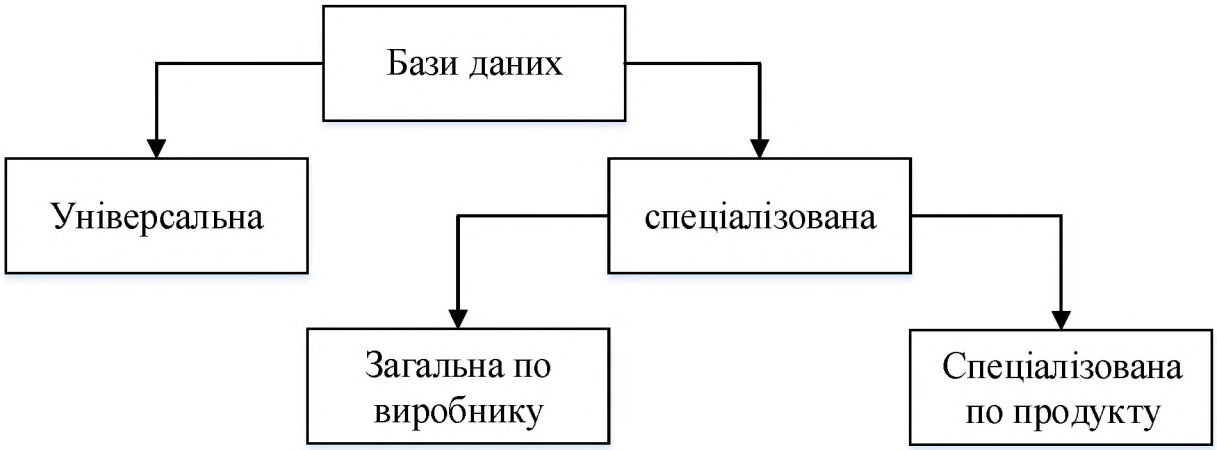

Рисунок 2.8 – Розподілення БД вразливостей за типом

Вивчення бази даних CVE [10] показало, що вона має значні зв'язки з іншими джерелами інформації про вразливості. Проте, ця база має недолік – відсутність специфікацій програмно-апаратного забезпечення, до яких відноситься виявлена уразливість.

Детальніші описи вразливостей знаходяться в базі даних NVD. Ця база містить інформацію про вразливості, пов'язані з програмно-апаратними продуктами, у форматі CPE. Крім того, NVD включає показники уразливостей у форматі CVSS (Common Vulnerability Scoring System) [10]. Однак кількість посилань на інші бази даних, які інтегруються з записами уразливостей, менше у базі NVD, ніж у CVE (остання включає всі посилання з NVD).

Слід зазначити, що NVD містить логічні описи конфігурацій програмно-апаратного забезпечення для кожної уразливості. Ці описи визначають вразливі системи у вигляді списків програмно-апаратного забезпечення, об'єднаних логічними операторами І / АБО. Конфігурація вважається вразливою, якщо містить логічний оператор І (тобто включає не менше двох продуктів одночасно).

Відмінність цих баз полягає у тому, що NVD є найбільш повною та структурованою. Тому її використовують як основу для співвіднесення записів уразливостей у інших базах.

Основні елементи структури записів уразливостей включають:

1. Статус – «Entry» (перевірено) або «Candidate» (ще не перевірено).

2. Фаза – етап розвитку уразливості і дата присвоєння цієї фази: Proposed, Interim, Modified, Assigned.

3. Опис – текстовий опис уразливості.

4. Посилання – адреса ідентифікатора джерела інформації про уразливість.

5. Голоси – імена членів голосування, які прийняли рішення про додавання уразливості до бази.

6. Коментарі – ім'я автора коментаря та його зміст.

Додавання уразливості до бази включає три етапи:

1. Обробку – аналіз, дослідження і форматування уразливості до стандарту CVE.

2. Присвоєння – призначення унікального ідентифікатора CVE.

3. Публікацію – створення нового запису та опублікування його на інтернет-ресурсі CVE, як тільки ідентифікатор CVE офіційно присвоєно.

Важливо зауважити, що більшість посилань з бази даних CVE належать до ресурсів інших баз вразливостей, а решта – до провідних виробників програмного та апаратного забезпечення.

Висновки, які можна зробити з результатів аналізу та статистики зв'язків із іншими джерелами, підтверджують, що база даних вразливостей CVE містить досить повний перелік уразливостей та має значну кількість посилань на бази вразливостей та виробників програмного й апаратного забезпечення. Проте, варто відзначити, що в базі CVE відсутній механізм опису приналежності вразливостей до конкретних продуктів, а також відсутність метрик та розрахунку ступеня небезпеки.

У травні 2012 року був запропонований формат «Загальна структура повідомлень про уразливість» (Common Vulnerability Reporting Framework – CVRF) [16]. З того часу відбувається процес переходу записів вразливостей CVE на цей формат. Основні відмінності CVRF від формату CVE включають:

– об'єднання полів «Опис», «Етап додавання» і «Фаза» в єдиному елементі «Запис» (Note);

– наявність у кожного запису уразливості порядкового номера (атрибут Order), який враховується з моменту створення бази CVE;

– можливість прив'язування записів вразливостей до списку продуктів, які потрапляють під цю уразливість.

Згідно з вищезазначеним, формат CVRF має свої переваги, проте найбільш вагомою є третя зміна.

2.4 Аналіз зв'язків джерел інформації бази CVE

База даних CVE, створена у 1999 році, налічує 294972 записи станом на 2023 рік. У таблиці 2.1 представлені бази даних, які були включені до списку вразливостей бази CVE та були активні на 2017 рік. Бази даних, які були вказані на сайті CVE як джерела, але відсутні у табл. 2.1, більше не функціонують. Основними причинами припинення роботи цих баз даних є банкрутство компаній, що стояли за їх створенням.

Таблиця 2.1 – Список джерел інформації CVE

Назва	URL адреса	Тип
AIXAPAR	http://www-01.ibm.com/support/search.wss	закрита
APPLE	http://lists.apple.com/archives/security-announce	відкрита
CERT	http://www.cert.org/advisories	відкрита
CERT-VN	http://www.kb.cert.org/vuls	відкрита
CHECKPOINT	http://www.checkpoint.com/defense/advisories/public/summary.html	відкрита
CISCO	http://www.cisco.com/en/US/products/products_security_ advisories_listing.html	відкрита
CONECTIVA	http://lwn.net/Alerts/Conectiva/	відкрита
DEBIAN	http://www.debian.org/security/	відкрита
EXPLOIT-DB	http://www.exploit-db.com	відкрита

На основі аналізу джерел CVE було складено таблицю 2.2, що містить короткий опис розглянутих баз даних. У цій таблиці вказано, чи доступ до цих баз є платним або безкоштовним для користувачів.

Таблиця 2.2 – Опис джерел CVE

Назва	Опис	Платна/ безплатна
AIXAPAR	Авторизований звіт про аналіз продукції IBM.	платна (Безкоштовна 30-денна версія)
APPLE	База вразливостей виробів apple	безкоштовна
CERT	Список вразливостей програмного забезпечення.	безкоштовна
CERT-VN	База даних Notes Vulnerability Notes містить відомості про вразливість програмного забезпечення. Примітки щодо уразливості включають резюме, технічні подробиці, інформацію про відновлення та списки зацікавлених постачальників. Більшість вразливих приміток є результатом приватної координації та розкриття інформації.	безкоштовна
CHECKPOINT	Поради щодо захисту від нападу та загрози доступні для сервісів Check Point Update Service та абонентів NGX SmartDefense.	безкоштовна
CISCO	Цей документ містить інструкції щодо отримання стабільного програмного забезпечення та отримання інформації про вразливість безпеки від Cisco.	безкоштовна
CONECTIVA	Цей документ містить список вразливостей продукції компанії Conectiva.	безкоштовна
DEBIAN	Ця сторінка показує становище Debian по відношенню до відомих вразливостей в системі.	безкоштовна
EXPLOIT-DB	База даних Exploit є CVE-сумісний архів публічних експлойтів і відповідного уразливого програмного забезпечення	безкоштовна
FEDORA	Архів що наповнюється інформацією розробниками та користувачами проєкту fedoraproject.org	безкоштовна
FREEBSD	Вебсторінка містить FreeBSD списку вразливостей та порад у їх вирішенні.	безкоштовна
GENTOO	Список вразливостей проєкту Gentoo Linux.	безкоштовна
JVNDB	База даних про захист від вразливості, зібрана на програмних продуктах, таких як операційні системи, додатки, бібліотеки та вбудовані системи, що використовуються в Японії.	безкоштовна
MANDRAKE	Список вразливостей Mandrake Linux.	безкоштовна
MS	Список оновлень безпеки Microsoft.	безкоштовна
NETBSD	Список вразливостей операційної системи NetBSD .	безкоштовна
OPENBSD	Список вразливостей операційної системи OpenBSD .	безкоштовна
REDHAT	Список вразливостей програмного продукту проєкту Red Hat.	безкоштовна
SECTRACK	Вебсервіс що надає інформацію про вразливості.	платна (час пробної версії не вказаний)
SECUNIA	Загальна база вразливостей.	безкоштовна

Інформація з табл. 2.2 спрощує процес пошуку вразливостей та їхнього усунення у випадку спрямованого пошуку. Також у таблиці бази даних з відміткою

«платна» є програмними продуктами, які використовують бази даних вразливостей для перевірки системи користувача та надсилають сповіщення про виявлені вразливості. Це зменшує необхідність користувача у самостійному моніторингу своєї системи.

2.5 Отримання та уточнення інформації в базі NVD

База даних NVD [11] отримує первинну інформацію з вебсайту CVE та послідовно аналізує її для визначення ключових показників впливу (CVSS), типів вразливостей (CWE), та інших відповідних метаданих. Оновлення інформації в NVD відбувається під час змін у вебсайті CVE або при запиті від виробників та інших зацікавлених сторін. У процесі запиту надається інформація, яку аналітики NVD перевіряють.

NVD не активно проводить власне тестування вразливостей, покладаючись на виробників та сторонні джерела. Після цього проводяться додаткові дослідження, щоб підтвердити, що CPE відповідають стандартам CPE [11] та включити їх до офіційного словника CPE. Якщо додаткова інформація стає доступною, результати та параметри CVSS можуть змінюватися.

Для деяких вразливостей повна інформація, необхідна для оцінки параметрів CVSS, може бути недоступною. Це зазвичай відбувається, коли виробник оголошує про вразливість, але відмовляється надавати певні подробиці. У таких випадках аналітики NVD присвоюють результатам CVSS найбільш консервативні оцінки. Так, якщо виробник не надає жодної інформації про вразливість, NVD призначить рейтинг 10.0 (максимальний рівень).

Відкриті бази вразливостей дозволяють виконувати пошук за різними параметрами на власних вебресурсах. Однак для здійснення детального пошуку за різними параметрами потрібно працювати з базами даних самостійно. Відомі

відкриті бази вразливостей надають можливість завантаження власної бази у декількох різних форматах. Наприклад, NVD підтримує формати XML та JSON, а база CVE доступна у форматах XML, TXT, CVRT, JSON та HTML.

2.6 Способи обробки інформації в XML документі

XML є одним із найпоширеніших форматів для відкритих баз вразливостей. Робота з XML-документами вимагає використання різноманітних програм – аналізаторів. Процес аналізу (або розбору) означає отримання потоку символів з вхідного документа та побудову внутрішньої структури, що відповідає поданій структурі. Результатом аналізу є модель XML-документа в пам'яті. Сам XML-документ є текстовим файлом, де деякі теги містять інформацію, а інші визначають структуру цієї інформації. Програма, яка зчитує та інтерпретує цю інформацію, називається аналізатором, а процес, під час якого інформація перетворюється із XML-документа в інший формат, – це перетворення.

Існує багато аналізаторів, які відповідають одному з встановлених стандартів: простому API для XML (SAX) або об'єктній моделі документа (DOM). Одним із способів обробки XML є використання мови XQuery.

Стандарт простого API для XML (SAX) був розроблений членами списку розсилки XML-DEV, це відкритий стандарт, не належить жодному консорціуму, комітету по стандартизації, компанії чи приватній особі. SAX сам по собі не є аналізатором, а є специфікацією, яка визначає інтерфейс для аналізаторів, найпоширеніші з яких розробляються для Java-платформи. У SAX визначаються стандарти для класів об'єктів, що використовуються при створенні SAX-аналізаторів.

SAX-аналізатори, що використовують API SAX, призначені для обробки великих XML-документів. Вони читають документ від початку до кінця, розділяючи його на невеликі блоки (відомі як блоки), обробляючи по одному блоку за раз. Після обробки блоку тексту, SAX-аналізатор визначає, чи містить цей блок

XML-теги чи іншу інформацію. Якщо знайдено XML-тег, аналізатор порівнює його з інформацією, визначеною програмою перетворення, що може бути створена за допомогою мови XSLT або іншою мовою. Після перетворення інформація видаляється з пам'яті, і процес повторюється з наступним блоком. Це дозволяє SAX-аналізаторам обробляти великі XML-документи, не обмежуючись обсягом доступної пам'яті. Однак, це також означає, що аналізатор не може звернутися до частини документа, яка вже була оброблена, а також не може обробляти інформацію кілька разів.

SAX є набором інтерфейсів Java. Більшість його методів реалізовано переважно саме на цій мові. Зараз методи SAX також впроваджено в RERL, PHP та інші мови програмування.

Об'єктна модель документа (DOM), як і SAX, є стандартом, побудованим на API, який розробники використовують для створення XML-аналізаторів. На відміну від SAX, DOM повністю завантажує XML-документ у пам'ять, перетворюючи його у деревоподібну структуру та зберігаючи в пам'яті. Це дозволяє отримати доступ до будь-якої частини документа. DOM також дозволяє створювати новий XML-документ та редагувати існуючий. Проте недоліком DOM є те, що вона працює тільки з документами, що повністю зберігаються в пам'яті.

XML DOM забезпечує розробникам інтерфейс для створення, навігації, додавання, зміни або видалення частин XML-документів, що не залежать від платформи. Цей прикладний інтерфейс API надає ідентифікатори, які можна використовувати для доступу до об'єктів і методів, і ці ідентифікатори однакові незалежно від мови програмування, на якій створено програмне забезпечення для роботи з XML-документами. У специфікаціях DOM консорціуму W3C використовується мова опису інтерфейсів IDL (Interface Description Language). IDL – це стандарт, розроблений групою OMG (Object Management Group). Щоб забезпечити сумісність DOM з різними браузерами, W3C розробила специфікації, які об'єднуються в групу W3C DOM.

DOM забезпечує логічне представлення структури документа у пам'яті, схоже до моделі документа XPath. Весь XML-документ відображений як вузол

Document, що знаходиться у вершині дерева. Вузли документа у моделі DOM можуть бути різних типів, найпоширеніші з них – це Node і Element. Вузол Node є базовим типом для більшості об'єктів у DOM. Він може містити будь-яку кількість дочірніх вузлів, а також батьківський вузол, за винятком кореневого, де батьківський вузол не існує. Об'єкт вузла Element використовується для подання елементів XML, які можуть містити атрибути, текстові дані та інші елементи.

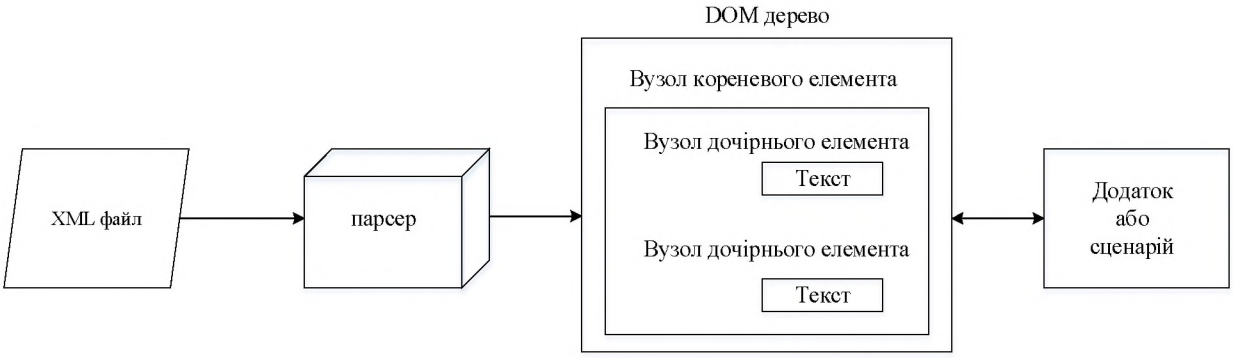

Рисунок 2.9 – Механізм використання DOM для роботи з XML документами

Мова XQuery (XML Query Language) дозволяє створювати запити до XML-документів, щоб отримати та обробити потрібну інформацію. У майбутньому передбачається розширення функціональності XQuery, включаючи можливості модифікації XML-документів – видалення, редагування тощо. XQuery використовує мову XPath для вибору фрагментів дерева документа.

Наразі активно розробляються різні реалізації мови XQuery, більшість з них має корпоративне призначення. Це свідчить про те, що значна кількість компаній вбачає у XQuery потужний інструмент.

Для ознайомлення з можливостями XQuery необхідне відповідне програмне забезпечення. Наразі існує декілька XQuery-процесорів, деякі є безкоштовними, а інші – комерційними програмами. У прикладах використовується XQuery-процесор Saxon. Цей програмний продукт постійно оновлюється, враховуючи нові підходи, розроблені корпорацією W3C. Розробник Saxon, Майкл Кей (Michael Kay), є членом робочих груп, які створили XQuery, XPath 2.0, XSLT 2.0. Однак, використання з командного рядка може бути незручним.

Запит XQuery завжди заключається у фігурні дужки {}. Він може містити умовний вираз, що визначає критерій пошуку. Результат виконання цього умовного виразу може бути істинним або ні. Якщо вираз є істинним для певного тегу, виконується відповідний запит.

Умовний вираз використовується всередині виразу FLWOR, який складається з речень for, let, where, order by і return. Речення for і let використовуються для привласнення значень змінним у межах XQuery.

Крім того, речення order by відповідає за сортування значень, а речення where застосовується для фільтрації даних. Речення let присвоює певне значення змінній, а речення return визначає, яку інформацію XQuery-процесор записує у вихідний файл.

Для виконання операцій з даними, часто потрібно перетворити інформацію в інші типи даних. Наприклад, текстові дані можна перетворити на числові. Для цього використовуються конструктори, які спрощують роботу з даними в XQuery.

Таблиця 2.3 – Деякі конструктори XQuery

Конструктор	**Опис**
xs:decimal	Створює із рядка десяткове число
xs:date	Створює із рядка дату за шаблоном YYYY-MM-DD (наприклад, 2009-12-21)
xs:double	Створює із рядка число з плаваючою крапкою подвійної точності
xs:float	Створює із рядка число з плаваючою крапкою
xs:int	Створює із рядка ціле число
xs:time	Створює із рядка час

У запитах XQuery можна використовувати такі арифметичні операції:

1) + для додавання,
2) – для віднімання,
3) * для множення,
4) div для ділення,
5) mod для визначення залишку від ділення.

Ці арифметичні операції застосовуються до числових даних типів: integer, decimal, float або double.

Також можна використовувати функції, такі як:

- sum для розрахунку суми послідовності значень,
- avg для розрахунку середнього значення,
- count для підрахунку кількості значень,
- max для визначення максимального значення у послідовності,
- min для визначення мінімального значення у послідовності.

Для звернення до значення атрибуту, як у мові XPath, використовується символ @.

Умовний оператор if визначає критерії пошуку інформації в XQuery і дозволяє виконати різні дії залежно від істинності логічної умови. Він має загальний вигляд, подібний до JavaScript:

if (умова) then {блок операторів 1}

else {блок операторів 2}

Механізм роботи оператору такий: якщо логічна умова виконується, виконується блок операторів 1, інакше – блок операторів 2.

У XQuery доступні різноманітні операції та функції, що спрощують обробку великих обсягів інформації про вразливості. Від арифметичних дій до розрахунку середнього значення та підрахунку кількості даних – ці можливості дозволяють ефективно аналізувати та використовувати отримані дані. Крім цього, наявність умовних операторів дозволяє створювати складні запити, які відповідають певним критеріям безпеки. Використання таких інструментів у XQuery стає важливою складовою для ретельного аналізу та обробки даних щодо вразливостей, сприяючи більш глибокому розумінню та виявленню ключових проблем безпеки.

Висновки до розділу 2

У другому розділі було проведено дослідження баз даних вразливостей та методів їх обробки. База даних CVE є загальною базою даних для вразливостей програмного забезпечення, яка містить унікальні ідентифікатори CVE для кожної

вразливості. База даних NVD є найбільш повною та структурованою базою даних вразливостей, яка містить інформацію про вразливості, пов'язані з програмно-апаратними продуктами, у форматі CPE. База даних NVD отримує первинну інформацію з вебсайту CVE та послідовно аналізує її для визначення ключових показників впливу (CVSS), типів вразливостей (CWE), та інших відповідних метаданих.

Оновлення інформації в NVD відбувається під час змін у вебсайті CVE або при запиті від виробників та інших зацікавлених сторін. Для деяких вразливостей повна інформація, необхідна для оцінки параметрів CVSS, може бути недоступною. У таких випадках аналітики NVD присвоюють результатам CVSS найбільш консервативні оцінки.

Відкриті бази вразливостей дозволяють виконувати пошук за різними параметрами на власних вебресурсах. Однак для здійснення детального пошуку за різними параметрами потрібно працювати з базами даних самостійно.

XML є одним із найпоширеніших форматів для відкритих баз вразливостей. Робота з XML-документами вимагає використання різноманітних програм – аналізаторів.

На основі отриманих висновків запропоновано наступні рекомендації щодо розробки систем виявлення та усунення вразливостей: для отримання інформації про вразливості слід використовувати відкриті бази вразливостей, такі як NVD або CVE; для обробки інформації з відкритих баз вразливостей слід використовувати аналізатори XML; для здійснення детального пошуку за різними параметрами слід використовувати власну обробку інформації з відкритих баз вразливостей. Впровадження цих рекомендацій дозволить підвищити ефективність систем виявлення та усунення вразливостей.

РОЗДІЛ 3
РОЗРОБКА ТА НАЛАШТУВАННЯ СКРИПТІВ APACHE SPARK ДЛЯ ОТРИМАННЯ ДАНИХ ВРАЗЛИВОСТЕЙ ТА ЇХ СТАТИСТИЧНОГО АНАЛІЗУ

3.1 Вибір оцінок вразливостей з системи CVSS

Спочатку розглянемо вплив вразливостей на доступність. Є дві основні категорії цього впливу (рис. 1.5):

– група «Базові показники» включає показник «вплив на доступність» (Availability Impact, A) зі значеннями: Відсутній (None, N), Частковий (Partial, P) та Повний (Complete, C);

– група «Показники середовища» визначає показник «вимоги до доступності AR» з трьома можливими значеннями: «низький» (low), «середній» (medium) або «високий» (high).

Також для оцінки та вибору вразливостей мають значення інші параметри:

– група «Базові показники» включає показник «вектор доступу» (Access Vector, AV) з варіантами: Локальний (Local, L), Сусідня мережа (Adjacent network, A), Мережевий (Network, N);

– група «Часові показники» містить показник «можливість експлуатації» (Exploitability, E) з варіантами: Неперевірений (Unproven, U), Доведено правильність концепції (Proof-of-Concept, POC), Функціональний (Functional, F), Високий (High, H), Не визначено (Not Defined, ND).

Оцінка «Базова формула» (BaseScore) від 0 до 10, класифікується як Низька (0 ... 4), Середня (4 ... 7) і Висока (7 ... 10).

Треба зауважити, що не всі репозиторії надають інформацію про групи «Часові показники» і «Показники середовища». Згідно зі стандартом CVSS[18], часова оцінка не перевищує базову і не менше на 33%, а оцінка середовища не перевищує часову оцінку, але знаходиться в межах від 0 до 10. Отже, на початковому етапі можна обмежитися лише базовими показниками.

Критерії вибору вразливостей розглянуто на прикладі вразливостей, що загрожують доступності вебсервісів. Обмеження для формування підмножин – вибірок вразливостей такі:

– виключно мережевий доступ (Network, N);

– частковий вплив на доступність (Partial, P);

– повний вплив на доступність (Complete, C).

Додатковою умовою фільтрації є структурна схема вебсервера, наприклад, DNS-сервер. Щоб проаналізувати невелику кількість вразливостей, можна використовувати розширений пошук в базі NVD. У розширеному пошуку на nvd.nist.gov дані поділені на сторінки по 20 записів на кожній (рис. 3.1, рис. 3.2).

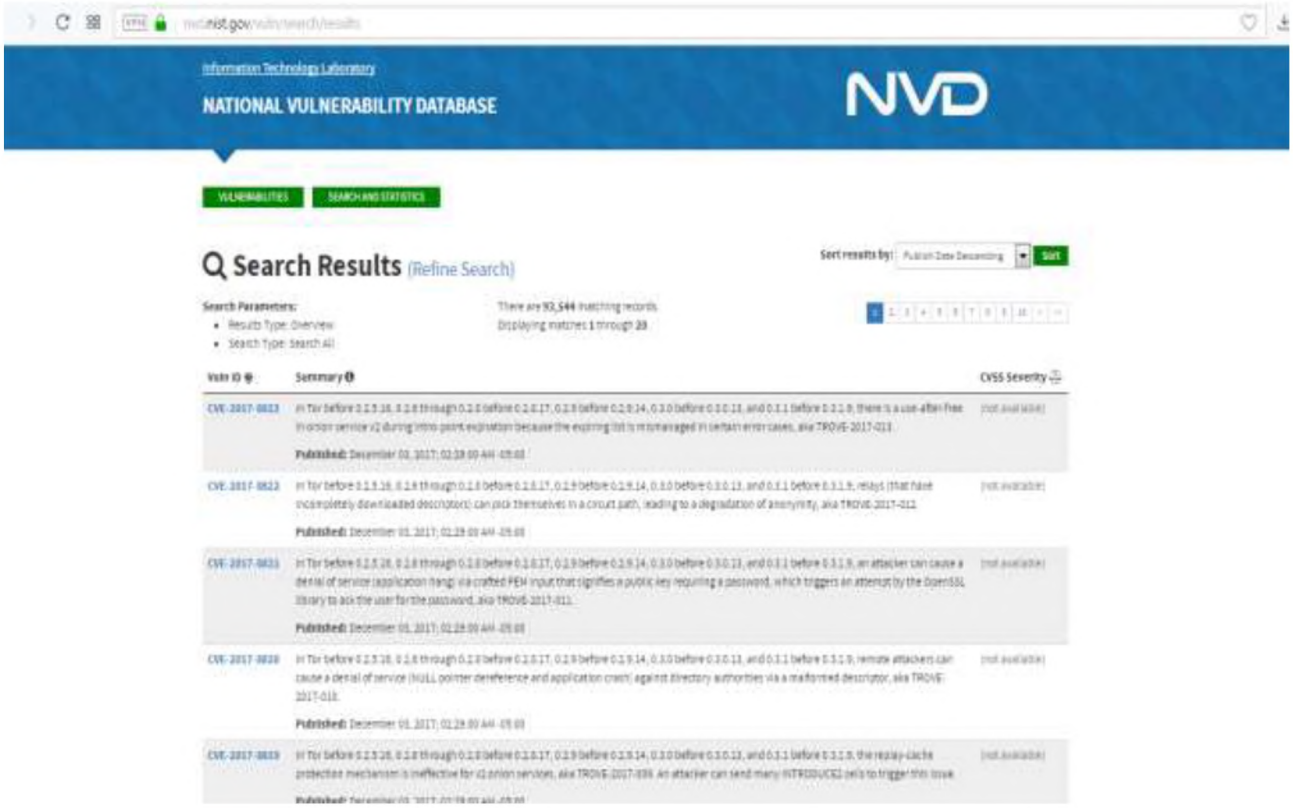

Рисунок 3.1 – Вебінтерфейс сховища вразливостей nvd.nist.gov

Отримання та обробка великих обсягів даних стає складним завданням і унеможливлює ручну обробку для досягнення поставлених цілей. Оптимальним рішенням для обробки бази вразливостей є використання JSON-документів. Це вимагає завантаження JSON-файлу відповідного року з сайту NVD. Для дослідження були використані дані з розділу «NVD / CVE JSON Feeds.

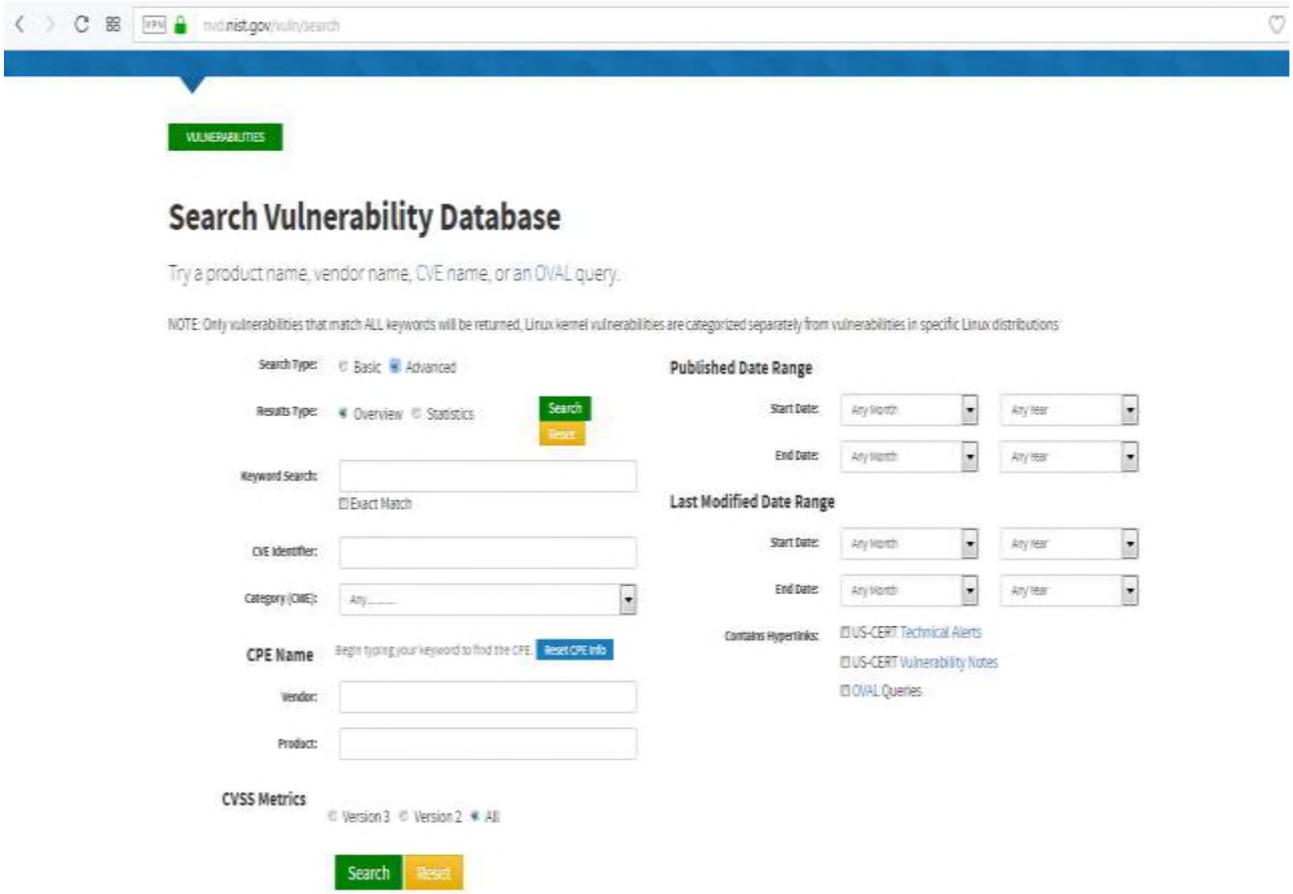

Рисунок 3.2 – Вебінтерфейс розширеного пошуку для формування підмножин вразливостей доступності

В епоху, коли дані стають ключовими, отримання і розуміння ідей та знань з величезних обсягів інформації вимагають відповідних інструментів та автоматизації. Один з таких методів – використання мов програмування, таких як Python, який використовується для аналізу даних та машинного навчання [5]. Крім того, інструменти, наприклад, Apache Spark і MongoDB, дозволяють обробляти великі набори даних у розподілених середовищах, забезпечуючи масштабованість і надійність [6, 7].

У цьому контексті ми використовуємо Python, Spark і MongoDB для аналізу CVE в операційних системах Ubuntu і Red Hat. Ми досліджуємо CVE за роками, порівнюємо їх вплив та виявляємо відмінності між останніми CVE та національною базою даних уразливостей (NVD). Ці інструменти автоматизації допомагають ефективно та масштабовано витягати ідеї та знання з цих даних.

Розроблені в ході роботи вихідні коди скриптів наведені в додатку А.

3.2 Скрипт для витягування даних з CVE

Скрипт на Python отримує дані про загальні вразливості та викладки (CVE) із сховища GitHub, оновлює локальну базу даних MongoDB. Скрипт зчитує конфігураційні параметри з файлу «cve_fetch.cfg», де зазначені локальний шлях до склонованого сховища GitHub [8], URI сервера та назва бази даних MongoDB, ім'я колекції MongoDB і частота оновлення даних.

Визначивши функцію «update_local_repo», скрипт автоматично оновлює локальне сховище. Якщо репозиторій існує, він виконує команду «git pull» для оновлення. У випадку відсутності репозиторію, скрипт клонує його. Функція повертає дату та час початку оновлення репозиторію.

Функція «user_option» відображає параметри оновлення локальної бази даних. Користувач може вибрати оновлення бази за допомогою повної колекції або останнього вилучення. Якщо обрано повну колекцію, скрипт розраховує час для отримання даних виходячи з дати та часу початку оновлення репозиторію.

Потім скрипт обробляє кожен JSON-файл у сховищі. Для кожного файлу перевіряється, чи відбулися зміни з часу оновлення репозиторію. Якщо так, скрипт зчитує дані JSON та вставляє чи оновлює документ у колекції MongoDB за відповідним _id даних CVE. У випадку помилки під час читання файлу, скрипт переходить до наступного.

Для читання конфігураційного файлу «cve_fetch.cfg» використовуються значення, що визначають локальний шлях до склонованого сховища GitHub, URI сервера MongoDB, назву бази даних MongoDB, назву колекції MongoDB і частоту оновлення даних.

3.3 Скрипт для витягування даних з NVD

Python-скрипт отримує дані про вразливості та експозиції (CVE) з національної бази даних уразливостей (NVD) та зберігає їх у базі даних MongoDB.

Спочатку скрипт імпортує необхідні модулі: MongoClient для підключення до сервера MongoDB, datetime і timedelta для роботи з часом, configparser для зчитування конфігурації з файлу, а також модулі для виконання HTTP-запитів до NVD API та регулярних виразів.

Параметри конфігурації зчитуються з файлу nvd_fetch.cfg за допомогою ConfigParser. Ці параметри включають URI сервера MongoDB, ім'я бази даних, назву колекції для використання, кількість днів для одноразового запиту, затримку між запитами та кількість результатів на сторінці.

Далі скрипт підключається до сервера та колекції MongoDB за допомогою об'єкта MongoClient. Він також встановлює URL-адресу та параметри для запиту до NVD API [9], включаючи кількість результатів на сторінці, початкову та кінцеву дати для запиту та початковий індекс для результатів.

Сценарій пропонує користувачеві вибрати параметр оновлення, такий як «повне» для оновлення всієї бази даних, дату у форматі ДД.ММ.РРРР для оновлення з певної дати або «вийти», щоб завершити роботу.

Далі скрипт виконує цикл з інтервалами днями, де кожен інтервал визначається кількістю днів з файлу конфігурації. Для кожного інтервалу він встановлює початкову та кінцеву дати для запиту API, обробляє сторінки результатів, вибирає та вставляє їх у колекцію MongoDB.

Якщо відповідь не містить даних, скрипт переходить до наступного інтервалу. Якщо ж дані отримано, скрипт перевіряє кожну вразливість, щоб визначити, чи вона вже присутня в колекції MongoDB. Якщо так, вона оновлюється новими даними, а якщо ні – додається новий запис. Після обробки всіх інтервалів скрипт завершує з'єднання з MongoDB та припиняє роботу.

3.4 Скрипти для знаходження параметрів вразливостей

Скрипт для знаходження частот згадувань вразливостей Ubuntu vs Red Hat використовує PySpark та Matplotlib для аналізу та візуалізації даних про кількість

вразливостей у CVE (загальні вразливості та експозиції). Спочатку код створює SparkSession з певними налаштуваннями для роботи з MongoDB. Далі він завантажує дані з MongoDB у фрейм даних PySpark, позначений як df.

Сценарій видобуває інформацію про рік із поля _id за допомогою регулярного виразу та функції regexp_extract. Потім розбиває стовпець «description_data» на рядки. Після цього він вибирає рік та значення з отриманих рядків, упорядковуючи їх за роком та створюючи два кадри даних для вразливостей Ubuntu і Red Hat. Операція фільтрації застосовується до вибору рядків, що містять «ubuntu» для кадру даних Ubuntu та «redhat» чи «red hat» для кадру даних Red Hat. Потім дані групуються за роками, обчислюється кількість вразливостей для кожного року за допомогою функції підрахунку.

Створюється новий фрейм даних за допомогою функції «range» для всіх років з 1999 по 2023. Цей фрейм даних зліва об'єднується («left join») з фреймами даних Ubuntu і Red Hat, включаючи роки з нульовою кількістю вразливостей. Остаточно отримані фрейми даних перетворюються на фрейми даних Pandas та сортуються за роками.

Matplotlib використовується для побудови графіків кількості вразливостей Ubuntu і Red Hat у вигляді лінійного графіка з роками на осі X та кількістю вразливостей на осі Y. Отриманий графік демонструє тенденцію кількості вразливостей для Ubuntu і Red Hat з плином часу.

Скрипт для розрахунку середніх оцінок для Ubuntu та Red Hat використовує Apache Spark для аналізу збережених у MongoDB даних про вразливості. Він витягує інформацію з MongoDB та перетворює її у фрейм даних Spark, відбираючи ключові стовпці для подальшого аналізу. Далі ці стовпці об'єднуються на основі поля «_id». Отриманий фрейм даних проходить фільтрацію, щоб включити тільки рядки, які містять «cpeMatch» з ключовими словами «ubuntu» або «redhat». Нарешті, скрипт розраховує середні значення «impactScore» та «exploitabilityScore» для кожної з операційних систем (Ubuntu і Red Hat) за допомогою систем оцінки CVSS (v2 і v3.1) і виводить результати на консоль.

У підсумку, цей скрипт проводить аналіз вразливостей у MongoDB за допомогою Spark, відфільтровуючи дані та розраховуючи середні оцінки для кожної можливої комбінації операційної системи та системи оцінки.

3.5 Скрипт для знаходження вразливостей CVE, не оброблених в NVD

Цей код використовує PySpark для обробки даних з двох колекцій MongoDB – cve.raw_cve та nvd.vulnerabilities. Він проводить аналіз, щоб знайти елементи CVE, які присутні у cve.raw_cve, але відсутні у nvd.vulnerabilities, та записує отримані результати до колекції processing.cve_not_in_nvd у MongoDB.

Початкова частина коду налаштовує сеанс Spark і визначає параметри конфігурації. Далі, він завантажує дані з обох колекцій MongoDB і перетворює їх у PySpark DataFrames: df_cve та df_nvd.

Код використовує метод віднімання (subtract), щоб виокремити значення стовпця _id з df_nvd DataFrame від df_cve DataFrame. Отриманий новий DataFrame df_subtracted містить лише значення _id, які є в df_cve, але відсутні у df_nvd. Наступний крок – фільтрація цього DataFrame за умовою року, більшого за 2021 (це визначено другим елементом поділу на _id).

У кінці, код розгортає масив description_data у df_cve DataFrame за допомогою функції explode, фільтрує результат, щоб включити рядки, де стовпець значень desc містить слово «linux» (незалежно від регістру). Він об'єднує цей DataFrame з df_subtracted DataFrame за допомогою стовпця _id, створюючи новий DataFrame df_filtered. Цей новий DataFrame містить лише значення _id, які присутні у df_cve, але відсутні у df_nvd та мають опис, що містить слово «linux». На завершення, код записує результат до колекції processing.cve_not_in_nvd у MongoDB за допомогою методу запису PySpark DataFrame.

Надалі, цей процес аналізу допомагає виявити та виділити конкретні вразливості, пов'язані з системою Linux, забезпечуючи можливість актуального та швидкого реагування на потенційні загрози. Такий підхід до обробки та визначення

вразливостей дозволяє зрозуміти динаміку їх поширення та виявлення, що створює основу для ефективних стратегій забезпечення безпеки в мережі.

3.6 Аналіз результатів за параметром частоти Ubuntu vs Red Hat

Аналізуючи випадки виявлення вразливостей у Ubuntu і Red Hat, помічаємо стабільне зростання їх кількості протягом тривалого періоду. Ця тенденція може бути пояснена розширенням та складнішим структуруванням програмного забезпечення, а також зростанням числа виявлених уразливостей завдяки вдосконаленню методів виявлення та процесів звітування. Графік частот зображено на рисунку 3.3.

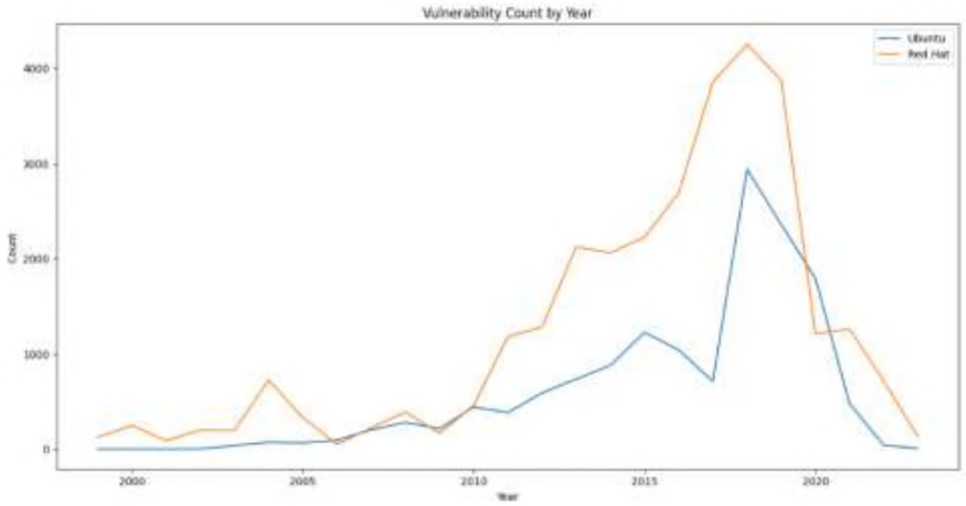

Рисунок 3.3 – Графік частот вразливостей Ubuntu та Red Hat

Незважаючи на те, що і Ubuntu, і Red Hat показують подібну експоненціальну тенденцію появи вразливостей, кількість останніх у Red Hat є майже вдвічі більшою, ніж у Ubuntu. Це можна пояснити тим, що Red Hat – це корпоративна операційна система, що використовується у різноманітних галузях та має широке застосування в бізнесі, тоді як Ubuntu в основному орієнтований на спільноту та використовується у особистих цілях та малому бізнесі.

Ще однією причиною, що може сприяти збільшенню кількості вразливостей у Red Hat, є його підтримка широкого спектру апаратних та програмних конфігурацій, що робить його більш складним та більш вразливим до потенційних атак. Більше того, Red Hat використовується у великих корпораціях, де потрібно більше налаштувань та інтеграції, що може підвищити ризик вразливості.

У той час як Ubuntu спроєктовано з урахуванням простоти та легкості використання, що робить його зручним та безпечним для особистого використання та невеликого бізнесу. Менша база користувачів та менш складна структура можуть призвести до меншої кількості вразливостей.

Узагальнюючи, більша кількість вразливостей у Red Hat порівняно з Ubuntu може пояснюватись поєднанням складності, конфігурацій та користувацької бази. Тим не менш, варто зауважити, що обидві операційні системи вразливі до ризиків та потребують постійного контролю та оновлень для збереження безпеки.

3.7 Оцінки впливу (impact) та використання (exploitability) v2 та v3.1

На підставі наданих даних можна виявити деякі відмінності між Ubuntu і Red Hat у середніх показниках CVSS та рівнях можливості використання.

Спочатку, аналізуючи середні оцінки CVSS v2, стає очевидним, що у Red Hat вищий середній показник впливу, ніж у Ubuntu. Це може бути пов’язано з рядом факторів, таких як відмінності в політиці або практиках безпеки обох компаній, типи вразливостей, що більш поширені в кожному дистрибутиві, або важливі різниці у важливості виявлених уразливостей. Графічне відображення отриманих середніх оцінок представлене на рисунку 3.4.

Крім цього, рейтинг використання у Red Hat також вищий, ніж у Ubuntu, що вказує на те, що вразливості, що виявляються в Red Hat, можуть бути легше експлуатовані, ніж у випадку з Ubuntu. Це може мати відношення до відмінностей у базовій структурі обох дистрибутивів або способу упаковки та поширення програмного забезпечення.

```
Ubuntu average impV2 score: 4.753415409055058
Ubuntu average impV31 score: 4.068043420704204
Ubuntu average expV2 score: 6.669843791369019
Ubuntu average expV31 score: 2.273259200423481
Red Hat average impV2 score: 5.067630356221321
Red Hat average impV31 score: 4.279409740147914
Red Hat average expV2 score: 7.414515573911572
Red Hat average expV31 score: 2.4542419549130265
```

Рисунок 3.4 – Отримані середні значення оцінок

Однак при аналізі середніх оцінок CVSS v3.1 ця тенденція змінюється: у Ubuntu виявлено незначно менший середній показник впливу та рівень зручності використання порівняно з Red Hat. Це означає, що різниця між оцінками зменшилася, у порівнянні з картинкою CVSS v2. Це може бути наслідком поліпшення практики безпеки Ubuntu протягом деякого часу, а також різниці в типах вразливостей, які виявляються в обох дистрибутивах.

Загалом важливо підкреслити, що відмінності в середніх показниках CVSS та можливості використання між Ubuntu і Red Hat є відносно невеликими, і ймовірно, обидва дистрибутиви мають міцні заходи безпеки для запобігання вразливостям. Однак ці тенденції можуть вказувати на області, де кожен дистрибутив може потенційно покращити свої заходи безпеки для зменшення впливу та можливості використання вразливостей у своїх системах.

3.8 Аналіз результатів вразливостей CVE необроблених в NVD

На момент написання цієї роботи було знайдено записи з бази CVE, які ще не були оброблені в NVD. Отримані записи було записано, згідно скрипту, в колекцію MongoDB. На рисунку 3.5 зображені отримані результати в представленні MongoDB Compass. За виявленими даними, в базі даних NVD ще залишилося п’ять записів, які ще не оброблені, і всі вони стосуються ядра Linux версії 6.2. Важливо відмітити, що ця версія є відносно новою і використовується лише у недавній версії Ubuntu 23.04.

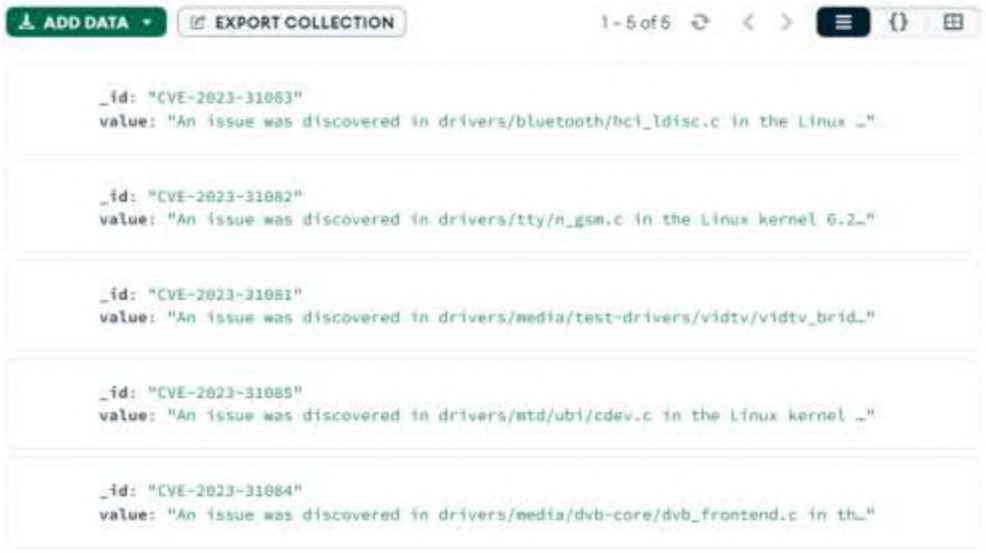

Рисунок 3.5 – Отримані дані з CVE, котрі ще не оброблені в NVD

Факт того, що NVD ще не опрацював ці п’ять записів, не обов’язково вказує на недбалість або затримку їхньої роботи. Між виявленням вразливостей та їхнім офіційним визнанням та документуванням зазвичай присутня певна затримка. NVD ґрунтується на інформації, наданій сторонніми постачальниками, дослідниками та громадами з відкритим кодом для виявлення та документування вразливостей. Отже, затримка у вирішенні цих п’яти записів може бути викликана дефіцитом інформації або ресурсів на даний момент.

Тим не менш, важливо визнати можливі ризики, пов’язані з цими п’ятьма невирішеними записами. Якщо ці вразливості не будуть вчасно визнані та усунені, вони можуть стати серйозною загрозою для безпеки систем, що працюють на ядрі Linux версії 6.2. Оскільки остання версія Ubuntu вже використовує цю версію ядра, слід уважно стежити за цими невирішеними записами та вживати необхідних заходів для захисту від можливих атак, поки NVD не зможе їх вирішити.

Крім того, ця ситуація підкреслює важливість наявності декількох джерел інформації у плануванні заходів з управління вразливостями. Організаціям слід спиратися не лише на NVD, а й відстежувати оновлення від постачальників програмного забезпечення, громад із відкритим кодом та дослідників у сфері безпеки. Це допоможе їм бути в курсі будь-яких виявлених уразливостей і вживати відповідних заходів для зменшення ризиків.

Підсумовуючи, хоча затримка у вирішенні цих п’яти записів у NVD може викликати певне занепокоєння, важливо залишатися пильними та активними у моніторингу вразливостей. Використовуючи різноманітні джерела інформації та вживаючи необхідних заходів, організації можуть знизити потенційний вплив будь-яких невиявлених уразливостей.

3.9 Розрахунок економічних затрат на розробку моделі безпеки інформаційної системи

Розрахунок витрат на розробку інформаційних системних моделей проведено згідно з [39]. Оцінка трудомісткості розробки моделей, розглянута як складова розробки програмного продукту, включає витрати часу на такі етапи:

– підготовку завдання;

– дослідження та створення алгоритму для розв’язання задачі;

– розробку блок-схеми алгоритму;

– програмування за готовою блок-схемі;

– відлагодження моделі в середовищі моделювання;

– підготовку відповідної документації;

– виявлення помилок та інше.

Загальну трудомісткість T_заг у людино-днях можна розрахувати за допомогою такої формули:

$$T_{заг} = N_{час} \cdot k_{скл} \cdot k_{м} \cdot k_{станд} \cdot k_{стандПП}$$
$$T_{заг} = 30 \cdot 1{,}08 \cdot 1 \cdot 0{,}7 \cdot 1{,}2 = 27{,}2 \approx 30 \text{ люд. днів} \quad (3.1)$$

де $T_{заг.}$ – загальна трудомісткість, вимірюється у людино-днях;

$N_{час}$ – норма часу, також у людино-днях (зазвичай: 30 людино-днів);

$k_{скл.}$ – коефіцієнт, що враховує проблеми контролю вхідної та вихідної інформації (рівень складності 1,08);

$k_{м.}$ – коефіцієнт, що відображає використання мови програмування різного рівня (високорівнева мова має коефіцієнт 1);

$k_{станд.}$ – коефіцієнт застосування стандартних програм (зазвичай 0,7);

$k_{станд.ПП}$ – коефіцієнт розробки стандартного програмного коду (значення змінюється від 1,2 до 1,6).

Під час розробки системи використовувався Apache Spark, отже, норму часу виправляють за допомогою коефіцієнта $k_{станд} = 0,6\text{-}0,8$.

Для визначення кількості розробників програмного продукту використовується формула (3.2), вимірювана у чисельності осіб.

$$Ч_i = \frac{T_{заг}}{Ф_{рч} \cdot \frac{t_{розроб}}{12}} \qquad (3.2)$$

$$Ч_i = 30/(249 \cdot \frac{1}{12}) \approx 1 \text{ чол.}$$

де $Ч_i$– кількість розробників моделей, особи;

$T_{заг}$ – загальна трудомісткість, людино-дні;

$Ф_{рч}$ – річний фонд робочого часу (для 2023 р. $Ф_{рч} = 249$ днів (цей показник щорічно змінюється), днів;

$t_{розроб}$ – запланований термін розробки, місяців.

Фактична вартість розроблених моделей обчислюється у процесі калькулювання собівартості, включаючи виробничу собівартість, адміністративні та витрати на збут.

Виробнича собівартість $С_{вир}$, грн., розраховується на основі поточних витрат на розробку (або функціонально-необхідних витрат на створення ПП)за формулою (3.3):

$$С_{вир} = С_{ЗП} + С_{ЄСВ} + С_{зв} + С_{м} \qquad (3.3)$$

$$С_{вир} = 9000 + 1430 + 7455,21 + 1491,04 = 19376,25 \text{ грн.}$$

де $С_{ЗП}$ – оплата праці розробника моделі, 9000 грн.;

$С_{ЄСВ}$ – єдиний соціальний внесок, грн. (з 01.01.2023 – 1430,0 грн);

$С_{зв}$ – загальновиробничі (накладні) витрати (розраховані в онлайн калькуляторі [39], 7455,21 грн);

$С_{м}$ – вартість матеріалів, комплектуючих, 1491,04 грн.

Окрім цих витрат на розробку моделей, витрати на створення та реалізацію програмного коду моделей включають:

– адміністративні витрати (організаційні, відрядження, страхування, амортизація, комунальні послуги, винагорода за послуги: юридичні, аудиторські, зв'язку, банку);

– витрати на збут (реклама, маркетинг, гарантійний ремонт, обслуговування, комісійні витрати);

– повна (фактична) собівартість (вартість індивідуальних витрат розробника коду моделей у конкретних умовах). Повна собівартість формується у бухгалтерському обліку витрат на розробку програмних моделей, використання ресурсів та технічного обліку.

Адміністративні витрати становлять 100·200 % від заробітної плати. Їх розраховують за формулою (3.4), грн:

$$С_{адмін} = k_{адмін} \cdot С_{ЗП}$$
$$С_{адмін} = 1{,}1 \cdot 9000 = 9900 \text{ грн.} \quad (3.4)$$

де $k_{адмін}$ – коефіцієнт адміністративних витрат.

Витрати на збут складають 2,5–5% від виробничої собівартості. Витрати на збут визначаються за формулою (3.5), грн:

$$С_{збут} = k_{збут} \cdot С_{вир.}$$
$$С_{збут} = 0{,}025 \cdot 19376{,}25 = 484{,}40 \text{ грн.} \quad (3.5)$$

де $k_{збут}$ – коефіцієнт витрат на збут ($k_{збут} = 0{,}025 \div 0{,}05$).

$С_{вир}$–виробнича собівартість, грн.

Також потрібно встановити частку кожного елемента витрат (питому вагу) у загальній сумі собівартості продукції у відсотках. Результати виконаних розрахунків занесено у таблицю 3.1.

Таблиця 3.1 – Калькуляція собівартості кінцевого продукту – програмного коду моделей безпеки інформаційної системи

Статті витрат	Сума, грн.	Питома вага, %
Заробітна плата	9000	30,2
Витрати на сплату єдиного соціального внеску	1430	4,8
Загальновиробничі (накладні) витрати	7455,21	25,1
Вартість витратних матеріалів, комплектуючих	1491,04	5,0
Виробнича собівартість ($С_{вир}$)	19376,25	
Адміністративні витрати	9900	33,3
Витрати на збут	484,4	1,6
Повна (фактична) собівартість ($С_{пов}$)	29760,65	100,0

У результаті проведеного аналізу витрат на створення та реалізацію програмного коду моделей, було виявлено, що значна частка витрат припадає на адміністративні витрати та незначна – на витрати на збут. Однак, заробітна плата, загальновиробничі витрати та витрати на матеріали також займають значне місце у загальній структурі витрат. Ці дані дають змогу управлінцям і фахівцям з бухгалтерського обліку краще розуміти складові витрат та ефективно планувати стратегії зниження собівартості продукту, спрямовані на оптимізацію фінансових ресурсів та підвищення конкурентоспроможності компанії.

Висновки до розділу 3

У розділі розглянуто методику розробки та аналізу моделей безпеки інформаційних систем. Запропоновано комплексний підхід, який включає такі етапи:

1. Збір та аналіз даних. На цьому етапі збираються дані про структуру, функціонування та особливості інформаційної системи, а також про ризики, яким вона піддається.

2. Розробка моделей. На цьому етапі розробляються моделі, які відображають особливості інформаційної системи та ризики, яким вона піддається.

3. Аналіз моделей. На цьому етапі аналізуються моделі з метою виявлення потенційних вразливостей та оцінки ризиків.

Для розробки моделей безпеки інформаційних систем запропоновано використовувати інструменти Python, Spark та MongoDB. Цей підхід дозволяє ефективно та масштабовано обробляти великі обсяги даних, а також створювати візуалізації для аналізу результатів.

Проведений аналіз вразливостей операційних систем Ubuntu і Red Hat дозволив сформувати наступні висновки. По-перше, Red Hat має більше повідомлень про вразливості, ніж Ubuntu, і це можна пояснити різними причинами, такими як відмінності у розмірі бази користувачів, популярності та моделях використання.

По-друге, зрозуміло, що Ubuntu і Red Hat мають різні профілі вразливості з різним ступенем серйозності. Схоже, що Red Hat має серйозніші вразливості з вищими оцінками CVSS порівняно з Ubuntu.

По-третє, наявність п’яти записів, які ще не оброблені в NVD, свідчить про можливу затримку в процесі оновлення. Ця затримка потенційно може зробити деякі системи вразливими до атак, тому користувачам важливо постійно оновлювати свої системи, щоб уникнути таких ризиків.

Таким чином, аналіз даних відкриває важливі відомості, які можуть стати основою для прийняття рішень щодо безпеки системи. Крім того, своєчасні оновлення мають вирішальне значення для забезпечення безпеки системи, а затримка в обробці цих записів підкреслює потребу в покращених системах оновлення, щоб мінімізувати ризики, пов’язані з невиправленими вразливими місцями.

ВИСНОВКИ

У роботі була поставлена і вирішена актуальна задача розробки програмного комплексу для автоматизації отримання кількісних характеристик вразливостей (часу прояву та критичності) на основі вибірки даних з відкритих баз.

1. Виконано аналіз сучасного стану галузі обчислень великих даних. У галузі big-data обчислень спостерігається стрімкий розвиток, обумовлений зростанням обсягів даних, що обробляються, та різноманіттям їх джерел. Для ефективного використання великих даних необхідно застосовувати спеціальні технології та методи, які постійно вдосконалюються.

2. У галузі опису вразливостей існує ряд стандартів, які дозволяють забезпечити уніфікований підхід до їх ідентифікації, оцінки та управління. Найважливішими з них є CVE, CVSS, CWE, CPE та CCE. CVSS є найбільш широко використовуваною системою оцінки вразливостей. Вона дозволяє оцінити вразливості за трьома основними групами метрик: базової, тимчасової та контекстної. Базова група відображає основні характеристики вразливостей, тимчасова – характеристики, що можуть змінюватися з часом, а контекстна – характеристики, що залежать від конкретного середовища. Використовуючи CVSS, можна визначити пріоритетність виправлення вразливостей, розташовуючи на першому місці ті, що представляють найбільшу загрозу.

3. База даних CVE є загальною базою даних для вразливостей програмного забезпечення, яка містить унікальні ідентифікатори CVE для кожної вразливості. База даних NVD є найбільш повною та структурованою базою даних вразливостей, яка містить інформацію про вразливості, пов'язані з програмно-апаратними продуктами, у форматі CPE. База даних NVD отримує первинну інформацію з вебсайту CVE та послідовно аналізує її для визначення ключових показників впливу (CVSS), типів вразливостей (CWE), та інших відповідних метаданих.

4. Оновлення інформації в NVD відбувається під час змін у вебсайті CVE або при запиті від виробників та інших зацікавлених сторін. Для деяких вразливостей повна інформація, необхідна для оцінки параметрів CVSS, може бути

недоступною. У таких випадках аналітики NVD присвоюють результатам CVSS найбільш консервативні оцінки.

5. Розглянуто методику розробки та аналізу моделей безпеки інформаційних систем. Запропоновано комплексний підхід, який включає такі етапи: збір та аналіз даних; розробка моделей та аналіз моделей. Для розробки моделей безпеки інформаційних систем запропоновано використовувати інструменти Python, Spark та MongoDB. Цей підхід дозволяє ефективно та масштабовано обробляти великі обсяги даних, а також створювати візуалізації для аналізу результатів.

6. Проведений аналіз вразливостей операційних систем Ubuntu і Red Hat дозволив сформувати наступні висновки. По-перше, Red Hat має більше повідомлень про вразливості, ніж Ubuntu, і це можна пояснити різними причинами, такими як відмінності у розмірі бази користувачів, популярності та моделях використання. По-друге, зрозуміло, що Ubuntu і Red Hat мають різні профілі вразливості з різним ступенем серйозності. Схоже, що Red Hat має серйозніші вразливості з вищими оцінками CVSS порівняно з Ubuntu. По-третє, наявність п’яти записів, які ще не оброблені в NVD, свідчить про можливу затримку в процесі оновлення. Ця затримка потенційно може зробити деякі системи вразливими до атак, тому користувачам важливо постійно оновлювати свої системи, щоб уникнути таких ризиків.

На основі отриманих висновків запропоновано наступні рекомендації щодо розробки систем виявлення та усунення вразливостей: для отримання інформації про вразливості слід використовувати відкриті бази вразливостей, такі як NVD або CVE; для обробки інформації з відкритих баз вразливостей слід використовувати аналізатори XML; для здійснення детального пошуку за різними параметрами слід використовувати власну обробку інформації з відкритих баз вразливостей.

Таким чином, поставлені задачі розв’язано у повному обсязі. Напрямком подальших досліджень є вдосконалення програмного забезпечення для автоматичного формування вибірок.