Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«_12_» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітній ступень бакалавр

освітня програма Кібербезпека
(назва освітньо-професійної програми)

на тему: Програмний засіб аналізу загроз в інформаційній системі

Виконавець: студент IV курсу, групи КБ-41

_____ **Ростислав Кравчишин**
(підпис) (ім'я, прізвище)

	Ім'я, прізвище	**Підпис**
Керівник	Олександр ЛАПТЄВ	

Нормоконтроль	Олександр ТОРОШАНКО	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студентові **КБ-41** (група) **Кравчишину Ростиславу Романовичу** (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Програмний засіб аналізу загроз в інформаційній системі

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Види кіберзагроз, типові кібератаки, властивості операційної системи Linux

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Аналіз загроз кібербезпеки, дослідити процес виявлення загроз, ознайомитись властивостями операційної системи GNU/Linux, розробити програмний засіб виявлення та аналізу загроз в інформаційній системі

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Підвищення ефективності аналізу загроз в інформаційній

системі за рахунок розроблення програмного засобу.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

Завдання видав _____ (підпис) Олександр ЛАПТЄВ (ім'я, прізвище)

Завдання прийняла до виконання _____ (підпис) Ростислав КРАВЧИШИН (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	24.10.2022 – 15.12.2022	*виконано*
2	Аналіз літератури	17.12.2022 – 13.01.2023	*виконано*
3	Обґрунтування вибору рішення	15.01.2023 – 29.01.2023	*виконано*
4	Аналіз загроз кібербезпеки	30.01.2023 – 17.02.2023	*виконано*
5	Дослідження процесу виявлення загроз	18.02.2023 – 30.02.2023	*виконано*
6	Розробка програмного засобу для аналізу загроз в інформаційній системі	1.03.2023 – 12.05.2023	*виконано*
8	Оформлення пояснювальної записки	13.05.2023 – 16.05.2023	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	17.05.2023 – 12.06.2023	*виконано*

Завдання видав

_____ (підпис) Олександр ЛАПТЄВ (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Ростислав КРАВЧИШИН (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 12 червня 2023 року

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, 1 додаток, має 56 сторінок основного тексту, 2 таблиці і 14 рисунків. Список використаних джерел містить 36 найменувань і займає 3 сторінки.

Методи дослідження кваліфікаційної роботи:

- аналіз відкритих джерел;
- аналіз механізмів захищеності та ризиків;
- моделювання загроз в інформаційній системі.

Метою дослідження є підвищення ефективності аналізу загроз в інформаційній системі.

Об'єктом дослідження процес аналізу загроз в інформаційній системі.

Предметом дослідження в даній роботі є метод аналізу загроз в інформаційній системі.

Практичною цінністю є підвищення ефективності аналізу загроз в інформаційній системі за рахунок розроблення програмного засобу.

Ключові слова: Аналіз загроз, безпека,операційна система GNU/Linux, мережевий трафік, C++, джерела загроз, збір даних.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ПЗ	–	Програмне забезпечення
ОС	–	Операційна система
IP	–	Internet Protocol
MAC	–	Media Access Control
DoS	–	Denial-of-Service
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol

ЗМІСТ

ВСТУП

Актуальність даної роботи полягає в тому, що зростаюча кількість загроз інформаційній безпеці ставить під загрозу ефективну роботу сучасних інформаційних систем. Аналіз загроз є важливим етапом в захисті інформаційних ресурсів, оскільки дозволяє виявити потенційні небезпеки та вжити заходів для їх запобігання або нейтралізації.

Розробка програмного засобу для аналізу загроз в інформаційній системі є актуальною з двох основних причин. По-перше, забезпечення захисту інформаційних ресурсів є надзвичайно важливим завданням для бізнесу, урядових установ, організацій та приватних осіб. Необхідність виявлення загроз та прийняття заходів для їх захисту стає все більш актуальною в умовах швидкого розвитку цифрової сфери.

По-друге, широке поширення операційної системи GNU/Linux у різних сферах, включаючи серверні середовища, хмарні рішення та вбудовані системи, робить розробку програмного засобу для аналізу загроз у цій операційній системі додатково актуальною. Забезпечення захисту в Linux-середовищах вимагає спеціалізованих рішень, які можуть аналізувати, виявляти та реагувати на потенційні загрози.

Таким чином, розробка програмного засобу аналізу загроз в інформаційній системі на базі операційної системи GNU/Linux є актуальним завданням, спрямованим на підвищення рівня безпеки інформаційних ресурсів та ефективного реагування на загрози в сучасному цифровому середовищі.

Тому **метою роботи** є підвищення ефективності аналізу загроз в інформаційній системі.

Для досягнення поставленої мети необхідно вирішити такі *завдання***:**

- Провести аналіз загроз кібербезпеки, моделі їх класифікації та пріоритизації;
- Дослідити процес виявлення загроз;
- Розробити програмну реалізацію ефективного аналізу та ідентифікації загроз в інформаційній системі.

Об'єктом дослідження в даній роботі є процес аналізу загроз в інформаційній системі.

Предметом дослідження в даній роботі є метод аналізу загроз в інформаційній системі.

Методи дослідження кваліфікаційної роботи бакалавра:

- аналіз відкритих джерел;
- аналіз механізмів захищеності та ризиків;
- моделювання загроз в інформаційній системі.

РОЗДІЛ 1
АНАЛІЗ ЗАГРОЗ КІБЕРБЕЗПЕКИ

1.1 Загрози кібербезпеки

Кіберзагроза – будь-яка подія або явище, що може негативно вплинути на діяльність організації, корпоративні активи, працівників, інші організації або на державу через використання інформаційної системи шляхом несанкціонованого доступу, втрати, неправомірного розкриття, зміни інформації або навіть відмови у доступі [1].

Джерела загроз визначають, звідки можуть походити загрози, тобто які конкретні сутності, фактори або обставини можуть стати джерелами потенційних загроз.

Основні джерела загроз можна описати більш детально наступним чином:

- Намір та метод, спрямовані на використання вразливостей: Це включає особливу намірену діяльність, спрямовану на зловживання вразливостями в інформаційних системах або мережах. Це може охоплювати напади на системи шляхом використання розроблених атак або зловживання вразливостями, що були розкриті.
- Ситуація та метод, які можуть випадково використовувати вразливості: Це включає непередбачувані ситуації або дії, які можуть ненавмисно використати вразливості системи. Наприклад, помилки в програмному забезпеченні, некоректна конфігурація або незнання правил безпеки можуть призвести до ненавмисного використання вразливостей.

Загалом, типи джерел загроз включають:

- Кібератаки – активності з боку зловмисників, спрямовані на злам інформаційних систем або фізичний доступ до об'єктів. Це може включати хакерські атаки, фішинг, зловмисний код, фізичні злами і т.д.

- Людські помилки в результаті ненавмисних дій або умисних дій – це помилки або недбалість працівників, які можуть призвести до компрометації безпеки. Ненавмисні дії можуть включати неналежну обробку даних, неправильну конфігурацію системи або незнання правил безпеки. Умисні дії можуть включати викриття конфіденційної інформації, крадіжку даних або недозволений доступ.
- Структурні вади організаційних ресурсів - вразливості апаратного забезпечення, програмного забезпечення або недостатній захист мережі. Наприклад, несправність апаратури, недостатня безпека програмного забезпечення або неправильна конфігурація можуть породжувати ризики для безпеки організації.
- Природні та штучно створені катастрофи – події та явища, що перебувають поза контролем організації: Це можуть бути природні стихійні лиха, такі як землетруси, повені, пожежі, а також штучно створені катастрофи, техногенні аварії або вимкнення енергопостачання. Ці події можуть призвести до недоступності систем або пошкодження інфраструктури, що може вплинути на безпеку даних та операцій організації.

1.2 Класифікація загроз

Класифікація загроз – це процес групування загроз за спільними характеристиками або критеріями. Вона служить для систематизації та упорядкування різноманітних загроз, що можуть впливати на інформаційну систему або організацію.

Класифікація загроз має декілька цілей:

- Розуміння загроз: класифікація допомагає зрозуміти різні типи загроз та їх характеристики. Це дозволяє краще усвідомити потенційні небезпеки та ризики, з якими можуть зіткнутися інформаційні системи;
- Аналіз вразливостей: класифікація загроз допомагає виявити, які конкретні вразливості можуть призвести до появи загрози. Це дає можливість більш ефективно спрямовувати заходи зі зміцнення безпеки та запобігати можливим атакам;

- Розробка заходів захисту: класифікація дозволяє визначити специфічні заходи захисту, які варто впровадити для запобігання різним типам загроз. Вона надає орієнтир для розробки стратегій та політик безпеки, які враховують специфіку конкретних загроз.

STRIDE – це методики класифікації загроз за кінцевою ціллю. Вона включає 6 типів основних загроз: підробка, порушення цілісності, заперечення, розголошення інформації, відмова в обслуговуванні, підвищення привілеїв[2].

У порівнянні з іншими моделями для класифікації загроз, модель STRIDE має свої власні переваги і недоліки.

Переваги моделі STRIDE:

- Простота та прозорість: STRIDE використовує простий та зрозумілий набір категорій загроз, що дозволяє легко ідентифікувати та класифікувати загрози;
- Фокус на типах загроз: STRIDE розглядає конкретні типи загроз, такі як підміна, втручання, відмова та інші, що допомагає зрозуміти їхні наслідки та визначити відповідні заходи захисту;
- Інтеграція з іншими моделями: STRIDE може бути використана в поєднанні з іншими моделями, такими як DREAD або CVSS, для більш комплексної оцінки ризиків.

Недоліки моделі STRIDE:

- Обмежена область застосування: STRIDE концентрується на ідентифікації і класифікації загроз, але не надає детального аналізу інших аспектів безпеки, таких як фізична безпека або соціальна інженерія.
- Відсутність кількісної оцінки: STRIDE не надає кількісної оцінки загроз або ризиків, що може бути корисним для порівняння та приоритезації.

1) Підміна (Spoofing) – загроза, яка полягає в підміні ідентичності або перехопленні облікових даних для отримання несанкціонованого доступу до системи[3]. Прикладами є:

- Підміна IP-адреси;
- Підміна MAC-адреси;
- Підміа email-адреси;

- Підміна SSL-сертифіката[4];
- Підміна HTTP-cookie;
- Отруєння кешу DNS-сервера[5].

2) Несанкціонована модифікація (Tampering) – загроза, яка полягає в зміні даних або коду системи без належних повноважень. Наприклад:

- Зміна вмісту бази даних;
- Створення нового користувача;
- Зміна пароля привілейованого користувача:
- Зміна вмісту директорії Active Directory або LDAP-системи;
- Модифікація мережевих налаштувань;
- Модифікація файлу з довіреними SSL/TLS-сертифікатами;
- Модифікація виконуваних файлів та скриптів.

3) Заперечення (Repudiation) – загроза, яка спрямована на виконання несанкціонованих операцій в системі, що не передбачає надійного відстеження та журналювання подій[6]. Прикладом може бути модифікація або видалення лог-файлу.

4) Розголошення інформації (Information Disclosure) – загроза, яка полягає в отриманні несанкціонованого доступу до певних даних[7]. Прикладами можуть бути:

- Несанкціноноване читання або копіювання файлу, що містить хеші паролів користувачів;
- Несанкціоноване читання або копіювання файлу, що містить приватний ключ SSH, SSL/TLS, Wireguard, OTP [8];
- Несанкціоноване читання або копіювання баз даних без необхідних повноважень;
- Несанкціоноване читання або копіювання електронних листів, отриманих за допомогою IMAP або POP3 клієнта.

5) Відмова в обслуговуванні (Denial of Service) – загроза, що спрямована на перекриття доступу користувачів до системи або ресурсів шляхом перевантаження або збою системи[9]. Прикладами є:

- Ping-флуд – використовується для перенавантаження цільової системи шляхом надсилання великої кількості пакетів Echo Request (ехо-запити) по протоколу ICMP (Internet Control Message Protocol). Це спричиняє витрату мережевих ресурсів та може призвети до непридатності цільової системи;
- SYN-флуд – відправка великої кількості пакетів SYN по протоколу TCP (Transmission Control Protocol) цільовій систему, використовуючи уразливості в трьохетапному процесі рукопожаття. Це може виснажити ресурси цільової системи і призвести до перешкоджання встановленню TCP-з'єднань іншими користувачами;
- UDP-флуд - полягає у відправці великої кількості пакетів UDP (User Datagram Protocol) цільовій системі, що спричиняє витрату ресурсів і створює мережеву навантаженість;
- HTTP-флуд - використовується для перенавантаження веб-сервера шляхом відправки великої кількості запитів по протоколу HTTP (Hypertext Transfer Protocol) або HTTPS (Hypertext Transfer Protocol Secure)[10];
- DNS-ампліфікація - використовується для перенавантаження цільової системи шляхом надсилання підроблених запитів по протоколу DNS (Domain Name System) до відкритих DNS-серверів, що призводить до отримання відповідей більшого розміру, ніж початковий запит. Ці збільшені відповіді потім надсилаються на цільову систему, що призводить до перевантаження.

6) Підвищення привілеїв (Elevation of Privilege) – загроза, яка полягає в отриманні несанкціонованого доступу до ресурсів або привілеїв, які перевищують рівень доступу, призначений для користувача. Прикладами є:

- Віддалена автентифікація привілейованого користувача з невідомої IP-адреси. Це може відбуватись у випадках, якщо порушник дізнався облікові дані привілейованого користувача шляхом соціо-технічної атаки або атаки повного перебору;
- Добавлення нового користувача в групу привілейованих користувачів (sudo, wheel)[11];
- Використання приватних ключів SSH або SSL/TLS сторонніми користувачами;

Типи загроз STRIDE можна поділити за властивостями інформації, яким вони загрожують. У таблиці 1.2.1 наведена ця відповідність.

Таблиця 1.1

Відповідність між типами загрози та властивостями інформації, яким вони загрожують

Тип загрози за класифікацією STRIDE	Властивість інформації
Підробка (Spoofing)	Конфіденційність, цілісність
Несанкціонована модифікація (Tampering)	Цілісність
Заперечення (Repudiation)	Спостережність
Розголошення інформації (Information Disclosure)	Конфіденційність
Відмова в обслуговуванні (Denial of Service)	Доступність
Підвищення привілеїв (Elevation of Privilege)	Конфіденційність, цілісність

STRIDE-per-element – це методика аналізу безпеки програмного забезпечення, яка дозволяє детально розглянути загрози та вразливості на рівні окремих елементів програми. Вона розширює класифікацію загроз STRIDE і дозволяє провести більш глибокий аналіз безпеки програмного забезпечення[12].

Даний підхід, застосовує модель STRIDE до таких елементів системи:

- Зовнішні елементи – системи або об'єкти, які знаходяться поза межами аналізованої системи, але взаємодіють з нею. STRIDE-per-element дозволяє виявити загрози, пов'язані зі спробами атак на зовнішні елемнти, передачею неправдивих даних або використанням системи для несанкціонованих дій.

- Процеси – операції, які відбуваються всередині системи. Аналіз STRIDE-per-element дозволяє виявити загрози, пов'язані з помилками в обробці даних, недостатніми перевірками безпеки, недоумисним використанням привілеїв тощо.
- Сховища даних – системи, де зберігаються дані. STRIDE-per-element допомагає виявити загрози, пов'язані зі спробами несанкціонованого доступу до даних, модифікацією або видаленням даних, а також втратою чи пошкодженням даних.
- Потоки даних – представляють передачу даних між різними елементами системи. STRIDE-per-element дозволяє виявити загрози, пов'язані зі спробами перехоплення, модифікації або спотворення даних під час їх переміщення.

STRIDE-per-element базується на класифікації загроз STRIDE, але зосереджується на окремих елементах системи. Замість загального огляду безпеки всього програмного продукту, методика STRIDE-per-element вимагає розглядати кожен елемент окремо та виявляти потенційні загрози, які можуть вплинути на цей елемент.

Наприклад, при аналізі веб-додатку, можна розглянути кожен його елемент окремо, такі як автентифікаційний модуль, обробка введених даних, управління сесіями тощо. Для кожного з цих елементів можна виконати аналіз STRIDE-per-element і виявити можливі загрози безпеці, які впливають на ці елементи конкретно.

Цей підхід дає змогу розробникам краще розуміти, які елементи програмного забезпечення піддаються більшій загрозі і потребують пріоритетної уваги в плануванні та реалізації заходів безпеки. Також він допомагає зосередитися на конкретних уразливостях і виробити ефективні заходи мінімізації ризиків для кожного елемента програмного забезпечення окремо.

STRIDE-per-element може використовуватися під час розробки програмного забезпечення або оцінки його безпеки для ідентифікації потенційних проблем та розробки відповідних контрмір. Використання цього методу дозволяє забезпечити більш повну і глибоку безпеку програмного забезпечення та запобігти можливим атакам і порушенням безпеки.

На таблиці 1.2.2 наведена відповідність типів загроз та їх цілями.

Таблиця 1.2

Відповідність між типами загроз за класифікацією STRIDE та елементами системи, що являються цілями загроз

Елементи системи	Spoofing	Tampering	Repudiation	Information Disclosure	Denial-of-service	Elevation of privileges
Зовнішні елементи	+		+			
Процеси	+	+	+	+	+	+
Сховища даних		+		+	+	
Потоки даних		+		+	+	

1.3 Оцінка загроз

Оцінка загрози - це процес визначення та встановлення пріоритету загрозам для інформаційної системи або організації. Вона використовується для оцінки серйозності та потенційного впливу загроз на безпеку інформації.

Оцінка загрози має наступні цілі:

- Визначення пріоритету: Допомагає встановити пріоритет для керування загрозами. Це дозволяє ідентифікувати та фокусуватися на найбільш серйозних загрозах, які потребують негайної уваги та заходів для забезпечення безпеки інформації.
- Раціональний розподіл ресурсів: Допомагає розподілити обмежені ресурси, такі як час, гроші та кадри, на найважливіші аспекти безпеки. Оцінка загрози дозволяє ефективно використовувати ресурси, зосереджуючи їх там, де загроза є найбільш значущою.

- Аналіз ризиків: Допомагає визначити ризики, пов'язані з різними загрозами, та оцінити ймовірність їх виникнення та потенційні наслідки. Це допомагає приймати обґрунтовані рішення з питань захисту та реагування на загрози.

Оцінка загрози допомагає організаціям зрозуміти, які загрози є найбільш серйозними та потенційно шкідливими для їх інформаційної системи, та приймати належні заходи для їх управління та зменшення ризиків. Вона сприяє ефективному плануванню та прийняттю рішень в галузі безпеки інформації.

Модель DREAD є методом оцінки загроз, що використовується для визначення потенційної серйозності загроз інформаційній системі[13]. Акронім DREAD означає наступні показники:

- Пошкодження (Damage) відображає потенційний рівень шкоди, яку може завдати загроза. Визначення Damage допомагає оцінити вплив загрози на інформаційну систему або бізнес-процеси організації. Зазвичай, Damage оцінюється за допомогою шкали, де вказуються можливі наслідки. Оцінка Damage дозволяє встановити пріоритетність заходів захисту та алокацію ресурсів на ті загрози, які можуть мати найбільший негативний вплив. Вищий рівень Damage вказує на більш значущу загрозу, яка вимагає негайних заходів для запобігання або зменшення можливих наслідків;

- Відтворюваність (Reproducibility) відображає те, наскільки легко або складно використати певну загрозу, які навички і знання потрібні для її реалізації, доступність необхідних інструментів та ресурсів, а також чи існують публічні документи, які описують певну вразливість, пов’язану з цією загрозою. Оцінка Reproducibility допомагає визначити потенційну загрозу та ризик для системи або додатку. Якщо атака легко відтворюється, це може вказувати на високий ризик, оскільки зловмисники з легкістю можуть скопіювати або використати атаку в своїх цілях. З іншого боку, якщо атака складна відтворити, це може знизити загрозу, оскільки для її успішного виконання зловмисники повинні мати високі технічні навички або доступ до спеціалізованих ресурсів;

- Експлуатація (Exploitability) відображає те, наскільки легко або складно зловмисникам використовувати загрозу або знайти вразливість в системі. При оцінці

Exploitability, враховуються такі фактори, як складність зловмисника знайти вразливість, наявність публічної інформації про вразливість або атаку, використання спеціалізованих інструментів або технік, а також наявність експлойтів або доказу концепції. Оцінка Exploitability допомагає визначити потенційну загрозу та ризик для системи або додатку. Якщо вразливість легко експлуатується, це може вказувати на високий ризик, оскільки зловмисники можуть легко скористатись вразливістю і здійснити атаку. З іншого боку, якщо вразливість складно експлуатується, це може знизити загрозу, оскільки зловмисники повинні мати високі технічні навички або доступ до спеціалізованих ресурсів для успішного зламування системи;

- Вплив на користувачів (Affected Users) відображає те, скільки користувачів можуть бути постраждалими внаслідок використання загрози або вразливості. або систем, які можуть бути постраждалими. При оцінці Affected Users, враховуються такі фактори, як кількість користувачів, які мають доступ до системи або додатку, їх важливість для бізнесу або організації, а також можливі наслідки для цих користувачів в разі успішної атаки або використання загрози. Оцінка Affected Users допомагає визначити потенційний вплив загрози на користувачів та визначити ризик для бізнесу або організації. Якщо велика кількість користувачів може бути постраждала, це може вказувати на високий ризик, оскільки наслідки можуть мати серйозний вплив на роботу системи. З іншого боку, якщо кількість постраждалих користувачів обмежена, це може знизити загрозу, оскільки вплив на систему буде менш значущим;
- Виявлення (Discoverability) відображає те, наскільки легко виявити загрозу, яка може бути використана для атаки. Оцінка Discoverability включає аналіз доступності інформації про вразливість або загрозу, наявності відкритих деталей про атаку або використання, а також наявності публічної документації, довідок або знань про загрозу. Якщо загроза легко виявляється або існує публічна інформація про неї, це може збільшити загрозу, оскільки зловмисники можуть швидше знайти та використовувати її для атаки. Чим більше інформації доступно про загрозу, тим більша ймовірність її використання зловмисниками. Оцінка Discoverability допомагає

визначити, наскільки широко поширена або відома є загроза, і дозволяє організаціям зосередитися на тих, які можуть бути більш доступними для зловмисників.;

Кожен з цих показників оцінюється на числовій шкалі від 0 до 10, де 0 - незначний ризик, а 10 - надзвичайно великий ризик. Сума балів за всі показники дає загальний бал ризику.

Також окрім DREAD існують інші методики оцінки загроз, такі як:

- CVSS (Common Vulnerability Scoring System) – це стандартизована система оцінки і рейтингування серйозності програмних вразливостей. Вона надає числовий бал, що вказує на рівень серйозності вразливості на основі різних факторів, таких як можливість використання, вплив та складність. CVSS допомагає організаціям пріоритезувати свої заходи відповіді та ефективно розподіляти ресурси для вирішення вразливостей [14].
- NIST Cybersecurity Framework – це набір рекомендацій, найкращих практик та стандартів, розроблений Національним інститутом стандартів і технологій (NIST) з метою забезпечення кібербезпеки.

CVSS (Common Vulnerability Scoring System) - це стандарт, що використовується для оцінки важкості і потенційного впливу вразливостей на інформаційну систему. В контексті кіберзагроз, CVSS допомагає організаціям квантифікувати та порівнювати різні вразливості з метою прийняття обґрунтованих рішень щодо призначення пріоритетів у вирішенні цих проблем.

CVSS визначається як числова оцінка на шкалі від 0 до 10, де 0 означає незначну загрозу, а 10 – надзвичайно серйозну загрозу. Ця оцінка заснована на кількох метриках, таких як вразливість, доступність, вплив і складність експлуатації. Кожна метрика має свою вагу, яка враховується для визначення загального балу CVSS.

Оцінка CVSS допомагає організаціям розуміти потенційний вплив вразливостей на їхню інформаційну систему та приймати рішення щодо вирішення проблем. За допомогою CVSS можна визначити, як швидко вразливість може бути експлуатована, яка шкода може бути завдана та які заходи можна прийняти для зниження ризику.

CVSS також дозволяє організаціям порівнювати різні вразливості та визначати пріоритети у вирішенні проблем. Вищі бали CVSS вказують на більш серйозні вразливості, які потребують негайного виправлення або застосування заходів безпеки. Нижчі бали CVSS можуть вказувати на менш критичні вразливості, які можуть бути вирішені в межах планового обслуговування.

CVSS включає такі основні показники: базовий показник, темпоральний показник, показник середовища.

Базовий показник (Base Score) в CVSS використовується для оцінки серйозності вразливості без врахування контексту атаки. Він враховує трьох метрики:

- Вектор атаки (Attack Vector): Вказує, яким чином зловмисники можуть отримати доступ до вразливості.
- Складність атаки (Attack Complexity): Оцінює складність експлуатації вразливості з боку зловмисників.
- Вплив (Impact): Визначає потенційні наслідки використання вразливості. Цей параметр оцінюється відносно конфіденційності, цілісності та доступності даних.

Темпоральний показник (Temporal Score) в CVSS використовується для оцінки серйозності вразливості з урахуванням часових факторів і змінюваних умов. Він враховує такі метрики:

- Експлуатабельність (Exploitability): Вказує, наскільки легко зловмисники можуть експлуатувати вразливість.
- Рівень доступної інформації (Remediation Level): Вказує на наявність доступних заходів для усунення вразливості.
- Ступінь спостереження (Report Confidence): Вказує на рівень впевненості в оцінці вразливості.

Показник середовища (Environmental Score) в CVSS враховує особливості конкретного середовища організації і дозволяє зробити більш точну оцінку ризику вразливості. Він включає наступні компоненти:

- Важливість вразливості (Vulnerability Importance): Цей фактор визначає, наскільки критично вразливість є для організації. Він враховує можливі наслідки

використання вразливості та значущість пошкоджень, які можуть статися в результаті атаки.

- Доступність заходів захисту (Security Measures Availability): Цей фактор оцінює, наскільки ефективно організація виконує заходи захисту, які можуть запобігти атакам або пом'якшити їх наслідки. Він враховує наявність і правильність використання заходів, таких як міжмережні екрани, антивірусне програмне забезпечення, системи виявлення вторгнень тощо.
- Можливості компенсації ризиків (Risk Compensation Potential): Цей фактор оцінює, наскільки організація може здійснити компенсацію ризиків, пов'язаних з вразливістю, за допомогою інших технічних, організаційних або процедурних заходів безпеки. Він враховує можливості резервного копіювання, відновлення систем, швидкості реагування на інциденти та інші аспекти, що сприяють зниженню ризику.

NIST Cybersecurity Framework є методикою, яка може бути використана оцінки та управління кіберзагрозами. NIST Cybersecurity Framework складається з таких етапів:

- Визначення контексту: На цьому етапі проводиться розуміння структури організації, бізнес-процесів та інформаційних активів. Враховуються фактори, такі як розмір організації, її цілі та завдання, а також рівень критичності активів.
- Виявлення загроз: На цьому етапі ідентифікуються потенційні загрози, які можуть вплинути на інформаційну безпеку організації. Враховуються як внутрішні, так і зовнішні загрози, включаючи нові кіберзагрози та відомі уразливості.
- Оцінка впливу загроз: На цьому етапі визначається потенційний вплив загроз на організацію, її активи та процеси. Враховуються можливі наслідки, такі як фінансові збитки, порушення конфіденційності, цілісності або доступності даних, а також вплив на репутацію та нормальне функціонування організації.
- Виявлення вразливостей: На цьому етапі проводиться аналіз вразливостей, які можуть бути використані загрозами для атак. Враховуються технічні вразливості, недоліки в безпеці процесів та можливі вразливості в інфраструктурі.
- Оцінка ризику: На основі інформації, зібраної на попередніх етапах, проводиться оцінка ризику. Ризик оцінюється шляхом поєднання імовірності

виникнення загроз та впливу, який вони можуть мати на організацію. Оцінка ризику допомагає визначити пріоритети та прийняти рішення щодо управління ризиками.

- Управління ризиками: Останній етап включає прийняття рішень щодо управління ризиками. Це можуть бути запобіжні заходи, які мінімізують ризики, або відновлювальні заходи, які допомагають відновити функціонування після інциденту.

Висновки за розділом 1

В даному розділі було розглянуто важливі аспекти, пов'язані з загрозами в інформаційній безпеці, а також моделями STRIDE і DREAD, які допомагають в класифікації та пріоритизації загроз.

Модель STRIDE надає систематичний підхід до класифікації загроз. Кожна літера у скороченні STRIDE відповідає певному типу загрози: Spoofing (підміна), Tampering (підробка), Repudiation (відмова), Information Disclosure (розкриття інформації), Denial of Service (відмова в обслуговуванні) та Elevation of Privilege (підвищення привілеїв). Модель STRIDE дозволяє ідентифікувати і оцінювати різні типи загроз та вживати відповідних заходів для їх запобігання.

Модель DREAD визначає п'ять основних аспектів, що допомагають в оцінці загрози: Damage (шкода), Reproducibility (повторюваність), Exploitability (експлуатаційність), Affected Users (постраждалі користувачі) і Discoverability (відкритість). Ці аспекти дозволяють визначити масштаб і вплив загрози на інформаційну систему та спрямовують зусилля на запобігання і ліквідацію вразливостей.

Обидві моделі є корисними інструментами для аналізу та управління загрозами в інформаційній безпеці. Вони надають структурований підхід до класифікації і оцінки загроз, що допомагає організаціям і фахівцям забезпечувати ефективний захист інформаційних ресурсів та приймати обґрунтовані рішення щодо приоритетів і вжиття заходів для мінімізації ризиків.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ПРОЦЕСУ ВИЯВЛЕННЯ ЗАГРОЗ

2.1 Огляд процесу виявлення загроз

Процес виявлення загроз включає три етапи:

- Збір даних з різних джерел, таких як системні журнали (логи), журнали окремих служб та застосунків, мережевий трафік, файлова система тощо. Збір даних може включати автоматичне збирання і реєстрацію подій, використання моніторингу мережевого трафіку. Метою цього етапу є збір якомога більшої кількості даних, які потенційно містять інформацію про можливі загрози [16].
- Попередня обробка даних – процес об'єднання різних даних в загальну структуру або формат для подальшого аналізу. Це допомагає створити комплексне представлення подій і виявити залежності та взаємозв'язки між різними даними [17].
- Співвідношення даних – пошук зв'язків і залежностей між різними подіями або даними. Співвідношення даних допомагає виявити підозрілі дії або загрози, які не можуть бути виявлені окремо [18].

Ці три етапи - збір даних, агрегація даних і кореляція даних - є важливими складовими процесу виявлення загроз. Вони спільно допомагають зібрати, структурувати та проаналізувати великий обсяг даних, що дозволяє виявити підозрілі активності та потенційні загрози для безпеки системи.

2.2 Джерела збору даних

Перед тим, як збирати дані, необхідно визначити джерела для них. У операційній системі GNU/Linux існує кілька основних джерел збору даних для виявлення загроз (рис. 2.1)[19]. Деякі з них включають:

- Системні журнали (логи) - містять записи про різні події, які відбуваються в операційній системі Linux. Це можуть бути журнали ядра (kernel logs), журнали системних служб, журнали автентифікації тощо. Системні журнали можуть містити

корисну інформацію про активність, помилки, вразливості або підозрілі дії, які можуть вказувати на потенційні загрози[20];

- Журнали окремих служб та застосунків. Наприклад, веб-сервери (наприклад, Apache або Nginx) можуть мати свої журнали доступу або журнали помилок, бази даних можуть мати журнали запитів, а поштові сервери можуть мати журнали доставки. Журнали окремих служб та застосунків можуть містити важливу інформацію про взаємодію зовнішніх сутностей або про відповідні події, що відбуваються в рамках служби або застосунку.
- Мережевий трафік – може бути важливим джерелом інформації для виявлення загроз. За допомогою інструментів моніторингу мережі, таких як libtins, можна перехоплювати, аналізувати і записувати мережевий трафік. Мережевий трафік може містити дані про вразливості, атаки, підозрілу активність, недопустимі спроби доступу тощо[21].
- Системні сигнали - спеціальні повідомлення, які відправляються операційною системою або програмами для інформування про певні події або стан системи. Наприклад, сигнали про недоступність послуг, падіння системи або помилки. Моніторинг системних сигналів може допомогти виявити незвичайну або підозрілу активність[22].
- Зовнішні джерела інформації - зовнішні сервіси, репозиторії загроз, бази даних адрес фішингових сайтів, списки відомих зразків шкідливого програмного забезпечення, інформацію про нові вразливості або підозрілу активність в інших системах, YARA-правила. Ці джерела можуть надати додаткову інформацію про потенційні загрози, яка може бути використана для виявлення загроз у власній системі.

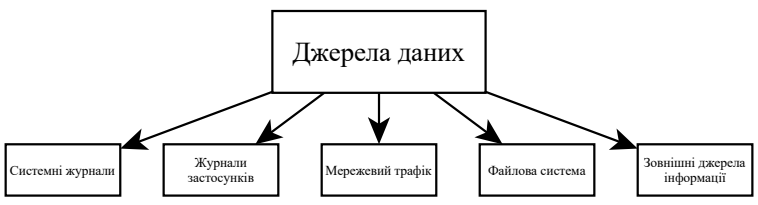

Рисунок 2.1 – Схема джерел даних для виявлення загроз

Журнали є невід'ємною складовою інформаційної безпеки та ефективного управління операційними системами та сторонніми застосунками. Вони виконують важливу роль у виявленні, аналізі та відповіді на події, що стосуються безпеки, а також в усуненні проблем та вдосконаленні роботи системи.

Журнали допомагають виявити та розслідувати події, пов'язані з несанкціонованим доступом, вторгненнями або незвичайною активністю в системі. Вони фіксують успішні та невдачні спроби аутентифікації, доступ до файлів, зміни в політиках безпеки та аккаунтах користувачів. Збирання цих даних дозволяє виявити ознаки атак та негайно реагувати на них, забезпечуючи високий рівень захисту інформації.

Окрім забезпечення безпеки, журнали допомагають в управлінні та відлагодженні системи. Вони реєструють важливі події, такі як запуск та вимкнення додатків, збої та важливі зміни конфігурації. Ці записи дозволяють адміністраторам системи виявити та вирішити проблеми, підвищуючи доступність та стабільність системи.

Крім того, журнали мають велике значення для виконання аудиту та відповідності. Вони надають докази виконання безпекових політик, регуляторних вимог та стандартів. Журнали забезпечують можливість відстежувати, хто, коли та яким чином отримує доступ до системи, змінює конфігурацію та взаємодіє з цінною інформацією.

Операційні системи для серверів, робочих станцій та мережевих пристроїв зазвичай реєструють різноманітну інформацію, пов'язану з безпекою. Найпоширеніші типи даних, пов'язаних із безпекою ОС, включають:

- Події системи. Події системи - це операційні дії, виконувані компонентами операційних систем, такі як вимкнення системи або запуск служби. Зазвичай, реєструються невдалі події та найважливіші успішні події, але велика кількість операційних систем дозволяють адміністраторам вказувати, які типи подій будуть реєструватися. Деталі, що реєструються для кожної події, також значно варіюються; зазвичай кожна подія має відмітку часу, а також інформацію, яка включає коди подій,

статус та помилки, назву служби, користувача або системний обліковий запис, пов'язаний з подією.

- Аудитові записи. Аудитові записи містять інформацію про події безпеки, такі як успішні та невдалі спроби аутентифікації, доступ до файлів, зміни політик безпеки, зміни облікових записів (наприклад, створення та видалення облікових записів, призначення привілеїв) та використання привілеїв. Зазвичай ОС дозволяють системним адміністраторам вказувати, які типи подій повинні бути аудитовані та чи слід реєструвати успішні або невдалі спроби виконати певні дії. В операційній системі Linux за журналювання відповідають програми syslog-ng, rsyslog або journald [23].

Журнали операційних систем також можуть містити інформацію від програмного забезпечення безпеки та інших програм, що працюють на системі. Журнали ОС найбільш корисні для виявлення або розслідування підозрілої діяльності, пов'язаної з певним хостом. Після виявлення підозрілої діяльності програмним забезпеченням безпеки, часто звертаються до журналів ОС, щоб отримати більше інформації про цю діяльність. Наприклад, мережевий пристрій безпеки може виявити атаку на певний хост; журнали операційних систем цього хоста можуть вказувати, чи був користувач увійшов у систему під час атаки і чи була атака успішною. Багато журналів в UNIX-подібних операційних систем створюються у форматі syslog [24].

Більшість організацій користуються різноманітними комерційними додатками, такими як поштові сервери та клієнти, веб-сервери та браузери, файлові сервери та клієнти для обміну файлами, бази даних та їх клієнти. Багато організацій також використовують різноманітні бізнес-додатки, які можуть бути комерційними або урядовими, такі як системи управління ланцюгом постачання, фінансове управління, системи закупівель, планування ресурсів підприємства та управління взаєминами з клієнтами. Крім комерційних та урядових додатків, більшість організацій також використовують спеціалізовані додатки, розроблені під їх конкретні вимоги.

Деякі додатки створюють власні журнали, тоді як інші використовують можливості журналювання операційної системи, на якій вони встановлені. Додатки

відрізняються значно в типах інформації, яку вони реєструють. Нижче наведено деякі найпоширеніші типи інформації, що реєструються, та потенційні переваги кожного з них:

- Запити клієнтів та відповіді сервера, що може бути дуже корисним при відновленні послідовностей подій та визначенні їх очікуваного результату. Якщо додаток реєструє успішні аутентифікації користувачів, зазвичай можна визначити, який користувач здійснив кожний запит. Деякі додатки можуть здійснювати докладне журналювання, наприклад, поштові сервери, які реєструють відправника, отримувачів, тему та назви вкладень для кожного електронного листа; веб-сервери, які реєструють кожний запит URL та тип відповіді, наданої сервером [25]. Деякі бізнес-додатки реєструють, які фінансові записи були доступні для кожного користувача. Цю інформацію можна використовувати для виявлення або розслідування подій та моніторингу використання додатків з метою відповідності і аудиту.
- Дані про облікові записи, така як успішні та невдачні спроби аутентифікації, зміни облікових записів (наприклад, створення та видалення облікових записів, назначення привілеїв), а також використання привілеїв. Крім виявлення подій безпеки, таких як спроба перебору пароля і підвищення привілеїв, це може використовуватися для визначення, хто і коли користувався додатком.
- Дані про використання, така як кількість транзакцій, що відбуваються за певний період часу (наприклад, хвилина, година) та розмір транзакцій (наприклад, розмір електронного листа, розмір передачі файлів). Це може бути корисним для певних видів моніторингу безпеки (наприклад, десятиразове збільшення активності електронної пошти може вказувати на нову загрозу, пов'язану з електронною поштою; надзвичайно великий вихідний лист може свідчити про некоректне розголошення інформації).
- Події життєвого циклу додатків, такі як запуск та вимкнення додатку, збої додатку та значні зміни конфігурації додатку. Це може бути використано для виявлення компрометації безпеки та неполадок в роботі.

Дані комерційних застосунків зазвичай передають дані по пропрієтарним протоколам, тому їх можна реєструвати тільки самими додатками, що робить журнали додатків особливо цінними для виявлення проблем безпеки, аудиту та відповідності. Однак, ці журнали часто мають пропрієтарні формати, що ускладнює їх використання, і дані, які вони містять, часто сильно залежать від контексту, що потребує додаткових ресурсів для їх перегляду та аналізу.

Велика кількість відслідковуваних журналів може призвести до проблеми управління журналами. Кожна система та програма може мати свої власні журнали, що призводить до розрізненості та розподілу журнальних даних по різним файлам. Це ускладнює аналіз та моніторинг подій, а також виявлення спільних залежностей та пов'язаностей між журналами різних систем. Крім того, формати та структура журналів також можуть відрізнятися. Різні системи та програми використовують різні формати журналів, що включають текстові файли з роздільниками, бази даних, syslog, SNMP, XML та інші. Це призводить до розбіжностей у способі збереження та подання журнальних даних. Внаслідок цього, аналіз та обробка журналів стає більш складним завданням, оскільки вимагає врахування різноманітних форматів та використання спеціалізованих інструментів для перетворення та обробки даних.

Моніторинг мережевого трафіку відіграє важливу роль у забезпеченні безпеки та виявленні мережевих загроз. Він дозволяє організаціям отримувати вичерпну інформацію про всі вхідні та вихідні пакети, що проходять через їхні мережі. Для цього використовуються спеціальні програмні рішення, які аналізують трафік і збирають відповідні дані.

Один з основних аспектів моніторингу мережевого трафіку - це аналіз протоколів, які використовуються для комунікації між пристроями. Крім базових протоколів, таких як TCP/IP, моніторинг може включати аналіз специфічних мережевих протоколів, таких як HTTP, FTP, DNS тощо. Аналіз протоколів дозволяє отримати важливі метадані про комунікацію, такі як джерело та призначення пакетів, порти, використовані протоколи, розмір пакетів тощо. Ці метадані можуть бути корисними для ідентифікації незвичайної або підозрілої активності в мережі.

Одним із головних аспектів моніторингу мережевого трафіку є забезпечення безпеки. Аналізуючи трафік, можна виявляти аномальну активність, таку як незвичайні запити, недопустимий обсяг даних або спроби несанкціонованого доступу. Моніторинг мережевого трафіку дозволяє вчасно виявляти потенційні загрози та інциденти безпеки, що допомагає убезпечити мережу та запобігти можливим атакам[26].

Однак, моніторинг мережевого трафіку також може постати перед викликами. Великий обсяг даних та потреба в аналізі метаданих можуть створювати складнощі у зборі, збереженні та обробці інформації [27]. Крім того, забезпечення конфіденційності інформації є ще одним важливим аспектом моніторингу мережевого трафіку, оскільки метадані можуть містити чутливу інформацію про користувачів та їхню взаємодію. Тому, при використанні моніторингу мережевого трафіку, важливо ретельно розробляти політики та процедури збирання та обробки даних, забезпечуючи відповідність нормам безпеки та конфіденційності.

Моніторинг мережевого трафіку включає аналіз різних типів мережевих пакетів, що мають важливе значення для безпеки. Розуміння цих типів пакетів допомагає виявляти потенційні загрози та вчасно реагувати на них.

Один з важливих типів пакетів, які аналізуються при моніторингу, це пакети з недостовірними джерелами. Це пакети, в яких джерело (IP-адреса або MAC-адреса) було підроблено або змінено з метою приховати справжнє джерело або спровокувати певну реакцію в мережі. Виявлення таких пакетів може вказувати на спроби зловмисників зламати мережеву безпеку або виконати атаку з маскуванням [28].

Інший важливий тип пакетів – це пакети з аномальними розмірами. Це пакети, які мають незвичайно великий або малий розмір, порівняно зі звичайними пакетами в мережі. Такі пакети можуть вказувати на спроби використання небезпечних атак, таких як переповнення буфера або використання надмірно великих даних для завантаження мережевого обладнання. Виявлення цих аномалій допомагає ідентифікувати потенційно шкідливі атаки і захистити мережу від них [29].

Також важливим типом пакетів є пакети з підозрілими або забороненими протоколами. Це пакети, які використовують протоколи або порти, які неочікувані

для даної мережі або не дозволені згідно з політиками безпеки. Наприклад, якщо у мережі діє політика блокування певних портів, то виявлення пакетів, які використовують ці порти, може свідчити про порушення політик безпеки або спробу обхіду захисту. Тому, моніторинг таких пакетів дозволяє виявляти потенційні атаки та реагувати на них [30].

Усі ці типи мережевих пакетів мають важливе значення для безпеки мережі. Аналіз та виявлення цих пакетів допомагає вчасно виявляти потенційні загрози, забезпечувати цілісність мережі та реагувати на можливі атаки. Це важлива складова частина мережевої безпеки і допомагає забезпечити надійність та безпеку мережевої інфраструктури.

2.3 Процес попередньої обробки даних

Метою попередньої обробки даних є приведення даних зібраних з різних джерел даних, що стосуються безпеки, до одного загального формату - події. Уніфікація дозволяє аналізувати взаємозв’язки та шаблони взаємодії між даними зібраними з різних джерел, що прямо не пов’язані між собою[31].

Подія – це об’єкт, що характеризує певні дані, зібрані під час виконання системи. Цей об’єкт містить інформацію про:

- Час виникнення події – відображає точний час або часовий проміжок, коли подія сталася або була зафіксована. Це може бути визначено з точністю до секунди, мілісекунди або інших одиниць вимірювання;
- Джерело, з якого була отримана інформація про подію. Наприклад, це може бути системний журнал, журнал окремої служби або застосунку, сенсор мережевої безпеки або інші джерела;
- Тип події – вказує на характер події, такий як доступ до файлу, спроба аутентифікації, мережевий запит, помилка аплікації тощо. Типи подій можуть бути задокументовані та класифіковані для полегшення аналізу та виявлення загроз;
- Додаткові дані – можуть бути додаткові атрибути або дані, які пов'язані з конкретним типом події. Наприклад, для подій аутентифікації це можуть бути

інформація про користувача, IP-адресу або тип аутентифікаційного протоколу. Для подій мережевого трафіку це можуть бути дані про порти, IP-адреси відправника та отримувача тощо. Додаткові дані залежать від типу події і можуть бути корисними для подальшого аналізу загроз.

Для подальшої обробки подій, їх потрібно зберегігати в одному сховищі даних, незалежно від джерела їх виникнення. Цей процес називається серіалізацією.

Серіалізація – це процес перетворення об'єктів у структурований формат, який може бути збережений, переданий або оброблений [32]. При серіалізації об'єкти перетворюються на послідовність байтів або інший текстовий формат, який може бути легко збережений у файлі, переданий через мережу або збережений у базі даних. Серіалізація подій має кілька важливих переваг:

- Збереження та передача даних: Серіалізовані події можуть бути збережені в файловій системі або базі даних для подальшого аналізу та збереження. Вони також можуть бути передані через мережу до інших систем або сервісів для обробки.
- Незалежність від мови програмування та платформи: Серіалізація подій дозволяє представити дані в універсальному форматі, що робить їх доступними для обробки різними мовами програмування та на різних платформах. Це спрощує інтеграцію і обмін даними між різними системами.
- Підтримка структурованих даних: Серіалізація дозволяє зберігати та передавати структуровану інформацію про події, включаючи час виникнення, тип події, джерело, додаткові дані та інше. Це допомагає зберігати контекст та деталі подій для подальшого аналізу та виявлення загроз.
- Ефективність та швидкість: Серіалізація може бути оптимізована для швидкої обробки великих обсягів даних. Серіалізовані дані можуть бути компактнішими та оптимізованими для швидкого збереження та передачі.

Процес серіалізації подій зазвичай включає в себе перетворення об'єктів у структурований формат, наприклад JSON або XML, за допомогою відповідних бібліотек або інструментів. У випадках, коли важлива продуктивність та швидкість виконання програми, використовується бінарна серіалізація, яка не потребує додаткових витрат ресурсів на парсинг даних. Ці серіалізовані дані потім можуть бути

збережені в файлі або передані до інших систем для подальшого аналізу та виявлення загроз.

Існує кілька основних видів серіалізації, які використовуються в залежності від платформи або потреби взаємодії з іншими системами:

- Бінарна серіалізація – полягає у перетворенні об'єктів або даних у бінарний формат, який зазвичай є компактним та ефективним для зберігання або передачі.
- Текстова серіалізація – полягає у перетворенні об'єктів або даних у текстовий формат, який може бути зрозумілим іншим системам та легко сприйматись людиною. Приклади включають серіалізацію у JSON (JavaScript Object Notation), XML (eXtensible Markup Language), CSV (Comma-Separated Values) або YAML (YAML Ain't Markup Language) формати.
- Серіалізація до бази даних – дозволяє зберегти стан об'єктів та їх взаємозв'язки для подальшого використання або відновлення. Під час серіалізації до бази даних, об'єкти або дані перетворюються в формат, який може бути збережений у таблицях бази даних. Кожен об'єкт або елемент даних зазвичай відображається в окремому рядку таблиці, а його атрибути або властивості зберігаються у відповідних стовпцях. При серіалізації до бази даних важливо визначити схему бази даних, яка відображає структуру та взаємозв'язки між об'єктами. Крім того, треба враховувати типи даних, обмеження та інші аспекти, які можуть впливати на збереження та відновлення даних.

2.4 Процес співставлення даних для виявлення загроз

Одним з ключових аспектів моніторингу безпеки є співставлення подій зібраних системою даних з різних джерел для виявлення потенційних загроз. Цей процес включає аналіз, порівняння та зв'язування інформації, що дозволяє отримати більш повне уявлення про події, які відбуваються в мережі або системі.

Співставлення подій дозволяє виявити складні шаблони або зв'язки, які можуть бути непомітними при аналізі окремих подій. Наприклад, виявлення незвичайної послідовності подій, таких як невдалий доступ до системи, після чого відбувається

зміна конфігурації або незвичайні мережеві запити, може свідчити про потенційну атаку або компрометацію системи. Співставлення подій допомагає виявити ці зв'язки та вжити відповідних заходів безпеки.

Для виявлення загрози, необхідно визначити:

- Джерела виникнення подій, що можуть становити загрозу;
- Типи подій;
- Часовий проміжок, в межах якого наявність та зв’язок певних подій можуть становити загрозу;
- Цілі атаки, частиною може бути потенційна загроза.

При співставленні подій важливо враховувати метадані, такі як часова мітка, джерело, тип події та інші атрибути. Ці метадані надають контекст та допомагають виявити незвичайні зв'язки або аномалії. Наприклад, співставлення підозрілої мережевої активності з інформацією про невдалі аутентифікаційні спроби або зміну привілеїв користувача може вказувати на можливий зловмисний доступ до системи.

Одним з ключових елементів процесу співставлення є визначення правил та шаблонів для виявлення загроз. Це включає розробку алгоритмів, які аналізують зібрані дані та порівнюють їх з заданими критеріями. Такі правила можуть базуватися на відомих сценаріях атак або аномальної поведінки, а також на статистичних методах аналізу.

Окрім того, важливо враховувати контекст при співставленні даних. Це означає, що необхідно брати до уваги особливості конкретної інформаційної системи, її архітектуру, конфігурації та типові шаблони поведінки користувачів. Наприклад, певні дії, які можуть здатися підозрілими у одній системі, можуть бути нормальними для іншої системи з різними правилами використання.

Під час співставлення даних також слід звернути увагу на взаємозв'язки між подіями. Це означає аналізувати послідовність та часові інтервали між подіями, а також їх взаємозв'язки у просторі (наприклад, мережеві з'єднання між різними вузлами). Такий аналіз допомагає виявити складні шаблони атак або координацію дій зловмисників.

Для ефективного співставлення даних важливо мати систему, яка забезпечує швидкий та потужний аналіз. Використання спеціалізованих технологій, таких як розподілені системи обробки даних або алгоритми машинного навчання, може значно покращити швидкість та точність виявлення загроз.

Усі ці аспекти входять у процес співставлення даних для виявлення загроз і допомагають підвищити ефективність та точність системи аналізу загроз в інформаційній системі. Правильно налаштований та програмно реалізований процес співставлення даних дозволяє оперативно виявляти загрози та приймати відповідні заходи для забезпечення безпеки системи.

Висновки за розділом 2

У цьому розділі був розглянутий процес виявлення загроз, що включає кілька етапів. Було визначено, що джерелами даних можуть бути системні журнали, журнали окремих служб та застосунків, мережевий трафік, файлова система тощо. Збір даних з цих джерел відбувається автоматизовано, що дозволяє отримати повну картину подій, що відбуваються в системі.

Далі, розглянута агрегація даних, яка полягає в об'єднанні та структуруванні зібраних даних з різних джерел. Цей процес допомагає встановити загальну картину подій та виділити важливу інформацію. Агрегація даних сприяє зрозумінню контексту та обсягу загроз, що допомагає зосередитися на найбільш важливих аспектах безпеки системи.

Одним з ключових етапів виявлення загроз є співвідношення даних, тобто встановлення зв'язку між подіями. Цей процес включає аналіз та порівняння даних для виявлення патернів, аномалій та зв'язків між різними подіями. Це дозволяє виявити потенційні загрози та прийняти відповідні заходи для їх запобігання чи нейтралізації.

Для ефективної обробки та збереження подій та загроз була розглянута серіалізація, що дозволяє перетворити структуровані дані в послідовність байтів або інших форматів, що можуть бути збережені в базі даних чи передані по мережі. Це

забезпечує зручний доступ до даних та їх збереження для подальшого аналізу та використання.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ АНАЛІЗУ ЗАГРОЗ В ІНФОРМАЦІЙНІЙ СИСТЕМІ

3.1 Опис програмного засобу

Розроблений програмний засіб для аналізу загроз в операційній системі GNU/Linux є інструментом, що забезпечує здатність збирати дані з різних джерел, агрегувати їх та виявляти потенційні загрози для безпеки системи.

Для виконання задач виявлення та аналізу загроз програмний засіб виконує моніторинг мережевого трафіку, спостерігає за діяльністю у файловій системі, аналізує журнали, виявляє зловмисні програми тощо. Для взаємодії користувача з даним застосунком розроблений графічний інтерфейс.

3.2 Структура програмного засобу

На рисунку 3.1 зображена схема структури програмного засобу.

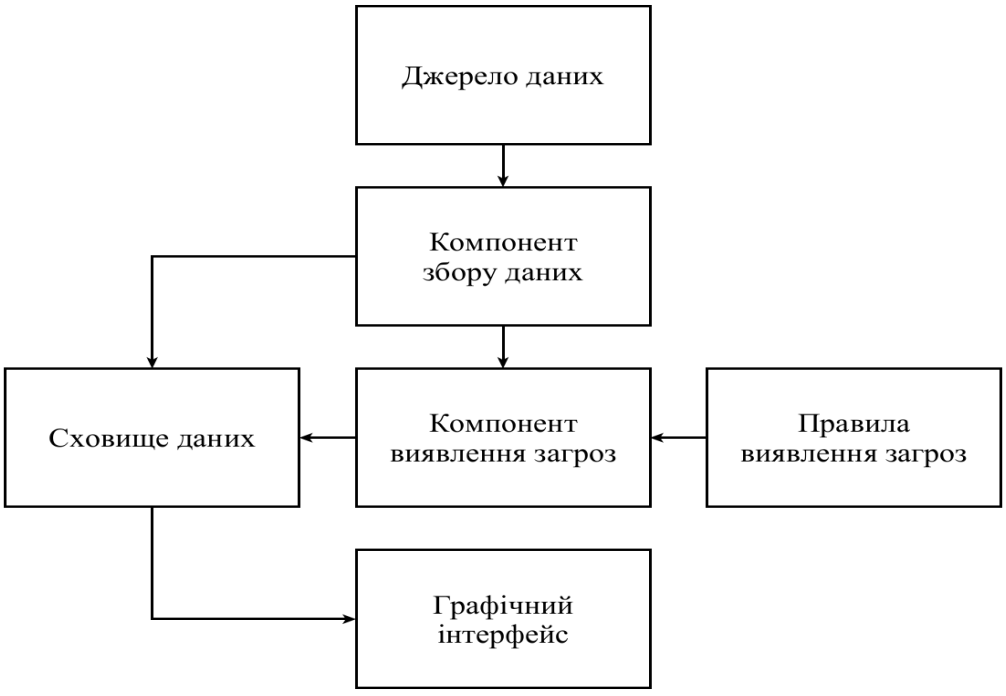

Рисунок 3.1 – Схема структури програмного засобу

3.3 Програмна реалізація

Для збереження даних програмний засіб використовує бібліотеку LevelDB. LevelDB забезпечує надійне та ефективне зберігання даних, використовуючи ключ-значення для організації даних у вигляді таблиці. Це дозволяє зручно і швидко зберігати та отримувати дані, необхідні для подальшого аналізу та обробки [32].

LevelDB - це високопродуктивна система управління базою даних типу “ключ-значення”, яка використовується для зберігання і управління великими обсягами даних. Одна з головних переваг LevelDB полягає у його ефективності і швидкості роботи. Дана бібліотека оптимізована для швидкого доступу до даних і виконання операцій збереження та отримання значень за ключем.

LevelDB володіє простим інтерфейсом, що робить його використання зручним для розробників. Він працює з ключами та значеннями у вигляді байтових масивів, що дозволяє зберігати будь-який тип даних. База даних підтримує операції додавання, оновлення та видалення даних, а також запити по ключу для отримання значень.

Ще однією важливою перевагою LevelDB є його компактність. Він забезпечує ефективне зберігання даних, що дозволяє економити місце на диску. Крім того, база даних підтримує стиснення даних, що дозволяє зменшити їх обсяг і забезпечити економію місця.

LevelDB є надійною та масштабованою базою даних. Вона вміє працювати з великими обсягами даних і забезпечує високу швидкість операцій незалежно від розміру бази даних. Крім того, LevelDB підтримує паралельну обробку, що дозволяє виконувати багатопоточні операції одночасно і підвищує продуктивність системи.

Для моніторингу мережевого трафіку програмний засіб використовує бібліотеку libtins. Ця бібліотека надає потужні інструменти для аналізу та перехоплення мережевих пакетів. З її допомогою можна отримати доступ до метаданих пакетів, таких як джерело, призначення, протоколи та інші характеристики. Це дозволяє програмному засобу виявляти аномальну або шкідливу активність в мережі та реагувати на потенційні загрози [34].

Libtins – це потужна бібліотека на мові C++, яка надає можливості моніторингу та маніпулювання мережевим трафіком. Вона дозволяє розробникам створювати власні програми для аналізу мережевих пакетів, перехоплення даних та взаємодії з мережевими пристроями.

Однією з ключових переваг Libtins є його простота використання та потужність. Бібліотека надає розробникам зручний інтерфейс для зчитування, редагування та створення мережевих пакетів. Вона підтримує різні мережеві протоколи, такі як Ethernet, IP, TCP, UDP, ICMP, ARP та інші, що дозволяє аналізувати та маніпулювати пакетами на різних рівнях стеку протоколів.

Ще одною перевагою Libtins є його кросплатформенність. Бібліотека підтримує різні операційні системи, такі як Windows, Linux та macOS, що дозволяє розробникам використовувати її на різних платформах без необхідності переписування коду. Це робить Libtins універсальним інструментом для розробки програм, пов'язаних з мережевим трафіком.

Для моніторингу подій у файловій системі програмний засіб використовує бібліотеку fanotify. Fanotify надає можливість відстежувати події, пов'язані з файловою системою, такі як відкриття, запис, зміна прав доступу та інші. Це дозволяє програмному засобу виявляти підозрілі або несанкціоновані зміни файлів, що може свідчити про потенційну загрозу безпеці.

Для збору інформації з журналів програмний засіб використовує бібліотеку liblognorm. Ця бібліотека дозволяє стандартизувати формат журналів, розпізнавати та аналізувати дані з різних журнальних джерел. Вона забезпечує можливість зручного та однорідного збору інформації з журналів, що спрощує подальший аналіз та виявлення загроз.

Liblognorm є бібліотекою, яка призначена для збору інформації з різноманітних журналів та їх нормалізації. Вона вирішує проблему розбіжностей у форматах журналів, дозволяючи однаково обробляти дані з різних джерел і приводити їх до єдиного стандартизованого вигляду. Це робить бібліотеку корисною для розробників систем моніторингу, аналітики журналів, систем виявлення вторгнень та інших програм, що працюють з журналами подій.

Графічний інтерфейс програмного засобу реалізовано з використанням бібліотеки Qt. Qt надає потужні інструменти для створення інтерактивних та привабливих графічних інтерфейсів. З його допомогою користувачі можуть взаємодіяти з програмним засобом, візуалізувати дані та отримувати зрозумілу та зручну для аналізу інформацію про виявлені загрози.

Для реалізації правил виявлення загроз програмний засіб використовує мову програмування lua. Lua є простою та гнучкою мовою, що дозволяє використовувати скрипти для визначення правил виявлення загроз. Використання lua надає гнучкість та можливість налаштовувати програмний засіб під конкретні потреби безпеки мережі.

Вхідною точкою програми є функція main. На рисунку 3.2 наведений код даної функції.

```
int main(int argc, char ** argv){
    leveldb::Options options;
    options.create_if_missing = true;
    leveldb::Status s = leveldb::DB::Open(options, "eventdb", &db);
    s = checkIfIndexPresent(db);
    if(!s.ok()){
        setEventIndex(db, 0);
    }


    QThread * capturePacketEventsThread = QThread::create(capturePacketEvents);
    QThread * storeEventsThread = QThread::create(storePackets);
    QThread * processEventsThread = QThread::create(processPackets);


    return 0;
}
```

Рисунок 3.2 – Код функції main

У даному фрагменті коду мови C++ відбувається виконання наступних дій:

- Створення об'єкту options типу leveldb::Options, який містить налаштування для роботи з базою даних LevelDB. Однією з опцій є create_if_missing, яка вказує, що якщо база даних не існує, то вона буде створена.
- Виклик функції leveldb::DB::Open, яка відкриває базу даних з використанням заданих налаштувань options і зберігає результат у змінну s типу leveldb::Status. Якщо відкриття бази даних успішне, то s буде мати значення ok().

- Перевірка, чи присутній індекс у базі даних. Це робиться шляхом виклику функції checkIfIndexPresent з передачею бази даних db. Результат перевірки зберігається у змінну s.
- Якщо індекс відсутній (тобто s не має значення ok()), то викликається функція setEventIndex, яка встановлює значення індексу для бази даних db.
- Створення потоків capturePacketEventsThread, storeEventsThread і processEventsThread, які викликають відповідні функції capturePacketEvents, storeEvents і processEvents. Це дозволяє виконувати ці функції в окремих потоках паралельно з основним потоком програми.
- Повернення значення 0 з функції main, що вказує на успішне завершення програми.

Ключовими класами у виконанні обробки подій та загроз є класи Event та Thread. Клас Event характеризує певну подію, що відбулась у системі. Ці події збираються та зберігаються у відповідному файлі, для подальшої обробки з метою отримання даних про потенційні загрози. можна було проаналізувати і в результаті отримати певні дані про потенційну загрозу. Ці дані характеризує клас Threat. Код даних класів наведений на рисунку 3.3 та на рисунку 3.4 відповідно. Класи Event та Threat є базовими та можуть бути наслідувані іншими класами, які можуть бути створені в завантажуваному модулі.

```
class Event {
public:
    char type[40];
    char title[256];
    std::time_t timestamp;
};
```

Рисунок 3.3 – Код класу Event

Поле type – це масив типу char з фіксованим розміром 40, який використовується для зберігання типу події. Його розмір обмежений 40 символами. Тип події зберігається у вигляді масиву символів з урахуванням використання похідних від Event класів, що додані у програмний засіб завантажуваними модулями. Символьний ідентифікатор типу дозволяє програмі динамічно визначати розмір об’єкту похідного класу або обробник цього об’єкту.

Поле title – це масив типу char з фіксованим розміром 256, який використовується для зберігання заголовка (назви) події. Його розмір обмежений 256 символами.

Поле timestamp – це змінна типу std::time_t, яка використовується для зберігання часової мітки події. Вона представляє кількість секунд, що пройшли з початку Епохи (зазвичай 1 січня 1970 року).

```
class Threat{
public:
    char type[40];
    std::streamoff offset;
    std::time_t startTime;
    unsigned int eventAmount;
    EventRef events[NUM_THREATEVENTS];
    uint8_t strideType;
    float dreadScore;
    char title[256];
};
```

Рисунок 3.4 – Код класу Threat

Поле type представляє тип загрози і зберігається у вигляді символьного масиву фіксованої довжини.

Поле startTime: Це поле визначає час початку виявлення загрози. Воно зберігає значення типу std::time_t, що дозволяє точно встановити момент часу, коли загроза була виявлена.

Поле eventAmount: Це поле вказує кількість подій, пов'язаних з даною загрозою. Воно відображає кількість виявлених подій, які потенційно вказують на наявність загрози.

Поле strideType визначає категорію загрози за класифікацією STRIDE. Воно представлено у вигляді беззнакового 8-бітного цілого числа (uint8_t), де кожне значення відповідає одній з категорій STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege). Ця змінна повинна містити значення однієї з констант заданих в перечисленні StrideType (рис. 3.5):

- STRIDE_SPOOFING: Ця константа відповідає загрозі "Спуфінг" і вказує на можливість підробки або псевдоідентифікації у системі;
- STRIDE_TAMPERING: Ця константа відповідає загрозі "Підробка" і вказує на можливість незаконного змінення даних або ресурсів у системі;
- STRIDE_REPUDIATION: Ця константа відповідає загрозі "Відмова від визнання" і вказує на можливість заперечення здійснених дій або передачі інформації у системі;
- STRIDE_INFORMATION_DISCLOSURE: Ця константа відповідає загрозі "Розголошення інформації" і вказує на можливість неправомірного доступу до конфіденційної інформації або розголошення її третім сторонам;
- STRIDE_DENIAL_OF_SERVICE: Ця константа відповідає загрозі "Відмова в обслуговуванні" і вказує на можливість перешкоджання нормальному функціонуванню системи шляхом перевантаження або блокування ресурсів;
- STRIDE_ELEVATION_OF_PRIVILEGE: Ця константа відповідає загрозі "Підвищення привілеїв" і вказує на можливість незаконного отримання додаткових привілеїв або прав доступу до системи.

Поле dreadScore: Це поле відображає оцінку загрози за моделлю DREAD. Воно представлено у вигляді числа з рухомою комою типу float.

Поле title – містить заголовок або опис загрози. Воно зберігається у вигляді символьного масиву фіксованої довжини і дозволяє надати короткий опис загрози для легшого розпізнавання та ідентифікації.

```
enum StrideType{
    STRIDE_SPOOFING,
    STRIDE_TAMPERING,
    STRIDE_REPUDIATION,
    STRIDE_INFORMATION_DISCLOSURE,
    STRIDE_DENIAL_OF_SERVICE,
    STRIDE_ELEVATION_OF_PRIVILEGE
};
```

Рисунок 3.5 – Код перечислення StrideType

Для того, щоб зберегти подію в базі даних, необхідно викликати функцію storeEvent (рис. 3.6). Ця функція виконує наступні кроки:

- Захоплює м'ютекс для забезпечення безпеки доступу до спільних даних;
- Додає отриманий об'єкт event до черги подій eventQueue;
- Звільняє м'ютекс, щоб інші потоки мали доступ до спільних даних.

Функція storeEvent використовує м'ютекс для синхронізації доступу до спільних даних та запобігання конфліктам у багатопотоковому середовищі. Це забезпечує безпечне додавання подій до черги для подальшої обробки. Використання м'ютексу дозволяє лише одному потоку виконувати критичну секцію коду одночасно, уникнувши проблем з гонками даних та некоректною обробкою інформації.

```
void storeEvent(Event *event){
    mutex.lock();
    eventQueue.push(event);
    mutex.unlock();
}
```

Рисунок 3.6 – Код функції storeEvent

Після цього події, що знаходяться в черзі обробляє функція storeEvents (рис. 3.7).

```
void storeEvents(){
    uint32_t i;
    getEventIndex(db, &i);
    while(1){
        sleep(1);
        if(eventQueue.empty()){
            continue;
        }
        Event * event = eventQueue.front();
        std::cout << "event: " << event->title << std::endl;
        std::cout << "timestamp: " << ctime(&event->timestamp)<< std::endl;
        std::cout << "size: " << event->size() << std::endl;
        std::cout << "index: " << i << std::endl;
        leveldb::Status s;
        leveldb::Slice key((char *) &i, 4);
        leveldb::Slice value((char *) event, event->size());
        s = db->Put(leveldb::WriteOptions(), key, value);
        eventQueue.pop();
        delete event;
        i++;
        setEventIndex(db, i);
    }
}
```

Рисунок 3.7 – Код функції storeEvents

У функції storeEvents відбуваються такі дії:

- Отримання значення індексу подій з бази даних за допомогою функції getEventIndex. Це дозволяє отримати поточне значення індексу, яке буде використовуватись для збереження подій;
- Виконання безкінечного циклу, що дозволяє продовжувати процес збереження подій неперервно;
- Затримка на 1 секунду за допомогою функції sleep. Це дозволяє знизити навантаження на систему та контролювати швидкість обробки подій;
- Перевірка, чи черга подій eventQueue не порожня. Якщо черга порожня, то виконується наступна ітерація циклу, очікуючи нові події для обробки;
- Отримання першого елемента з черги подій за допомогою front. Цей елемент представляє собою подію, яку потрібно зберегти;
- Виведення інформації про подію, такої як заголовок (title), час (timestamp), розмір (size) та індекс (i). Ця інформація друкується на екрані і дозволяє контролювати процес збереження подій;

- Збереження події у базі даних за допомогою функції leveldb::Put. Для цього створюються ключ і значення, які відповідають індексу та самій події. Потім вони передаються в функцію Put, яка здійснює збереження у базі даних;
- Видалення обробленої події з черги за допомогою pop. Це дозволяє видалити оброблену подію, оскільки вона вже збережена у базі даних;
- Збільшення значення індексу на 1 та оновлення його в базі даних за допомогою функції setEventIndex. Це дозволяє оновити значення індексу для наступних подій, які будуть зберігатись.

У даній програмі графічний інтерфейс відіграє важливу роль у взаємодії з користувачем і відображенні результатів аналізу подій і загроз. Програмний засіб використовує фреймворк Qt для підтримки графічного інтерфейсу. Qt надає широкий набір готових компонентів і інструментів для розробки інтерфейсу користувача, таких як вікна, кнопки, списки, таблиці та інші. Qt забезпечує механізми для організації взаємодії між елементами графічного інтерфейсу та логікою програми. Один з таких механізмів - це система "Сигнали та слоти", яка дозволяє зв'язувати події (сигнали), що виникають у віджетах, з методами (слотами), які будуть виконуватись відповідно до цих подій.

Клас ThreatDWindow є основним класом, відповідальним за головне вікно програми і управління графічним інтерфейсом (рис. 3.8). Даний клас успадковує від класу QMainWindow, що забезпечує базовий функціонал головного вікна програми. У ньому визначені різні компоненти інтерфейсу, такі як вкладки (QTabWidget), деревовидні списки (QTreeWidget), мітки (QLabel), кнопки (QPushButton), таблиці (QTableWidget) та інші. Ці компоненти використовуються для відображення інформації про події, загрози, залогованих користувачів та інші дані, а також для навігації між різними розділами програми.

```
class ThreatDWindow : public QMainWindow{
Q_OBJECT
public:
    explicit ThreatDWindow();
    void processEventGUI(std::streamoff * offset);
    void processThreatGUI(std::streamoff * offset);
    QTabWidget * tabWidget;
    QTreeWidget * threatsWidget;
    QTreeWidget * eventsWidget;
    QTreeWidget * loggedInUsersWidget;
    QWidget * sidePanel;
    QLabel * sidePanelLabel;
    QPushButton * sidePanelButton;
    QVBoxLayout * sidePanelLayout;
    QTableWidget * sidePanelTable;
    QScrollArea * threatsScrollArea;
    QScrollArea * eventsScrollArea;
    QScrollArea * loggedInUsersScrollArea;
    QScrollArea * sidePanelScrollArea;
    QSplitter * splitter;
    QTableWidgetItem * tableItems[15][2];
    char mac_addr[18];
    char ip_shost[16];
    char ip_dhost[16];
private slots:
    void handleEventSelections();
    void handleThreatSelections();
    void handleSidePanelButtonClick();
};
```

Рисунок 3.8 – Код класу ThreatDWindow

Клас ThreatDWindow відповідає за головне вікно програми і має різні компоненти і функціонал, пов'язаний з обробкою подій і загроз.

Основні елементи та функції класу ThreatDWindow включають:

- Конструктор ThreatDWindow : Цей конструктор ініціалізує об'єкт класу ThreatDWindow та його компоненти;
- processEventGUI та processThreatGUI: Ці функції відповідають за обробку подій і загроз у графічному інтерфейсі. Вони приймають вказівник на зміщення (offset) та виконують відповідні дії з подіями або загрозами;
- Різні компоненти інтерфейсу: Клас ThreatDWindow має різні об'єкти QTabWidget, QTreeWidget, QLabel, QPushButton, QVBoxLayout, QTableWidget, QScrollArea, QSplitter та інші, які використовуються для створення графічного інтерфейсу програми. Вони включають вкладки (tabs), деревовидні списки (tree widgets), елементи таблиці (table items) та інші елементи інтерфейсу;
- private slots: Цей розділ містить декларації приватних слотів. У Qt слоти (slots) – це спеціальні функції, які використовуються для обробки подій, взаємодії з користувачем і виконання певних дій у відповідь на виникнення певних подій. Ckjnb відповідають за обробку подій, що виникають у зв'язку з взаємодією з елементами

інтерфейсу. Наприклад, handleEventSelections обробляє події вибору подій, а handleThreatSelections обробляє події вибору загроз;

При ініціалізації об’єкта класу ThreatDWindow виконується його конструктор. На рисунках 3.9 та 3.10 наведений їхній код.

```
ThreatDWindow::ThreatDWindow(){
    tabWidget = new QTabWidget;
    threatsWidget = new QTreeWidget;
    eventsWidget = new QTreeWidget;
    loggedInUsersWidget = new QTreeWidget;
    sidePanel = new QWidget;
    sidePanelLabel = new QLabel;
    sidePanelButton = new QPushButton;
    sidePanelLayout = new QVBoxLayout;
    sidePanelTable = new QTableWidget();
    threatsScrollArea = new QScrollArea;
    eventsScrollArea = new QScrollArea;
    loggedInUsersScrollArea = new QScrollArea;
    sidePanelScrollArea = new QScrollArea;
    splitter = new QSplitter;

    QList<int> widgetSizes;
    widgetSizes.append(980);
    widgetSizes.append(300);
    QStringList headerLabels;
    headerLabels.push_back(QString("Назва"));
    headerLabels.push_back(QString("Час"));

    threatsWidget->setColumnCount(2);
    threatsWidget->setHeaderLabels(headerLabels);
    threatsWidget->setColumnWidth(0, 350);
    eventsWidget->setColumnCount(2);
    eventsWidget->setHeaderLabels(headerLabels);
    eventsWidget->setColumnWidth(0, 350);
    eventsWidget->setSelectionMode(QAbstractItemView::SingleSelection);
    threatsScrollArea->setWidget(threatsWidget);
    threatsScrollArea->setWidgetResizable(true);
    eventsScrollArea->setWidget(eventsWidget);
    eventsScrollArea->setWidgetResizable(true);
    loggedInUsersScrollArea->setWidget(loggedInUsersWidget);
    loggedInUsersScrollArea->setWidgetResizable(true);
```

Рисунок 3.9 – Перша частина коду конструктора класу ThreatDWindow

```
    tabWidget->addTab(threatsScrollArea, "Загрози");
    tabWidget->addTab(eventsScrollArea, "Події");

    sidePanelTable->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
    sidePanelTable->horizontalHeader()->hide();
    sidePanelTable->verticalHeader()->hide();
    sidePanelTable->setEditTriggers(QAbstractItemView::NoEditTriggers);
    sidePanelTable->setColumnCount(2);
    sidePanelTable->setRowCount(15);
    sidePanelLabel->setStyleSheet(QString("font-weight: bold; font-size: 14px"));
    for(int i = 0; i < 15; i++){
        for(int j = 0; j < 2; j++){
            tableItems[i][j] = new QTableWidgetItem();
        }
    }

    sidePanelLayout->addWidget(sidePanelLabel);
    sidePanelLayout->addWidget(sidePanelButton);
    sidePanelButton->setText("Показати події");
    sidePanelButton->setVisible(false);
    sidePanelLayout->addWidget(sidePanelTable);
    sidePanel->setLayout(sidePanelLayout);
    sidePanelScrollArea->setWidget(sidePanel);
    sidePanelScrollArea->setWidgetResizable(true);

    splitter->addWidget(tabWidget);
    splitter->addWidget(sidePanelScrollArea);
    splitter->setSizes(widgetSizes);
    this->resize(1280,720);
    this->setCentralWidget(splitter);

    connect(eventsWidget, SIGNAL(itemSelectionChanged()), this, SLOT(handleEventSelections()));
    connect(threatsWidget, SIGNAL(itemSelectionChanged()), this, SLOT(handleThreatSelections()));
    connect(sidePanelButton, SIGNAL(clicked()), this, SLOT(handleSidePanelButtonClick()));
}
```

Рисунок 3.10 – Друга частина коду конструктора класу ThreatDWindow

У конструкторі класу ініціалізуються різні елементи графічного інтерфейсу, такі як вкладки (QTabWidget), дерева (QTreeWidget), кнопки (QPushButton), направляючі (QVBoxLayout), таблиця (QTableWidget), прокрутні області (QScrollArea) та розділювач (QSplitter).

В конструкторі встановлюються розміри та параметри віджетів, встановлюються заголовки стовпців таблиць, встановлюються режими вибору елементів, створюються зв'язки між подіями (сигналами) та методами (слотами) за допомогою функції connect.

Зокрема, метод handleEventSelections() виконується, коли відбувається зміна вибору елементів у віджеті eventsWidget. Аналогічно, метод handleThreatSelections() виконується при зміні вибору елементів у віджеті threatsWidget. Метод handleSidePanelButtonClick() виконується при натисканні на кнопку sidePanelButton.

У результаті цього коду створюються необхідні віджети та їх налаштування для побудови графічного інтерфейсу програми ThreatDWindow.

Для прикладу було проведено моделювання загрози SYN-сканування портів. SYN-сканування портів є одним з популярних методів сканування мереж для виявлення відкритих портів на цільових системах. Цей вид сканування використовуєся для встановлення з'єднання з портом шляхом відправлення пакета з прапорцем SYN і очікування відповіді від системи. Процес SYN-сканування може дати користувачеві інформацію про те, які порти відкриті, а отже, доступні для зловмисника для подальшого експлуатації або атаки на систему.

Принцип роботи SYN-сканування полягає відправці пакетів з прапорцем SYN на різні порти системи, до якої здійснюється сканування. Якщо порт відкритий, система відправляє пакет з прапорцем SYN-ACK, що свідчить про готовність встановити з'єднання. У випадку, коли порт закритий, система відправляє пакет з прапорцем RST. За допомогою обробки отриманих відповідей можна визначити стан портів системи, виявити вразливості або незахищені сервіси, а також встановити обсяг доступної інформації про систему.

На рисунку 3.11 зображено вікно програми до виявлення загрози. На рисунку 3.12 у вкладці “Загрози”, можемо побачити одну виявлену загроза з назвою “SYN-

сканування”. Натиснувши кнопку “Показати події”, ми можемо переглянути події, що відносяться до цієї загрози – в даному випадку це TCP-пакети з прапорцями SYN та RST (рис. 3.13).

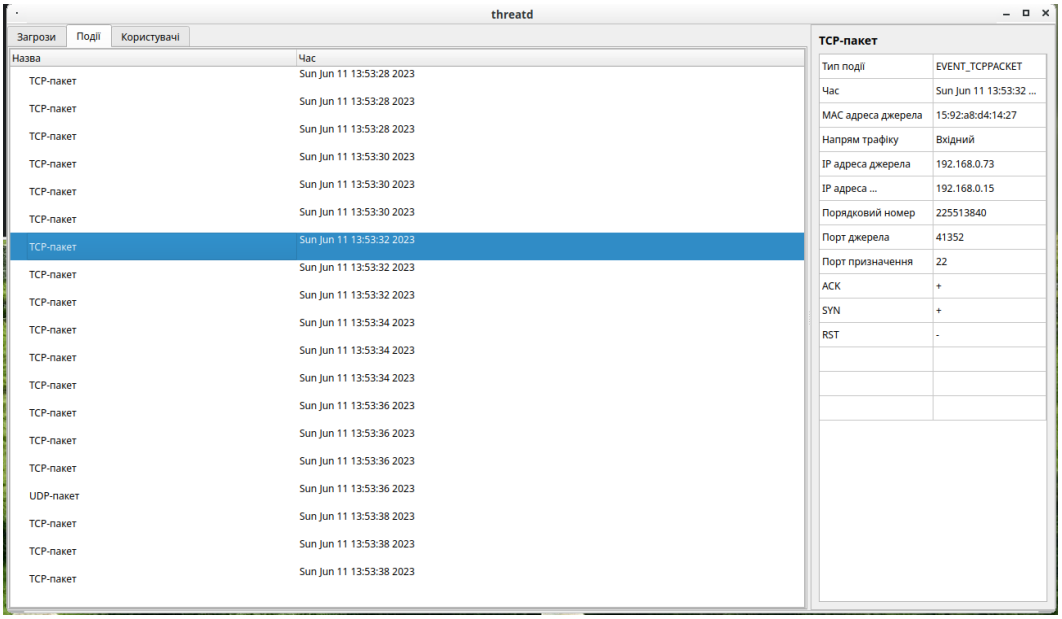

Рисунок 3.11 – Перегляд події у вікні програмного засобу

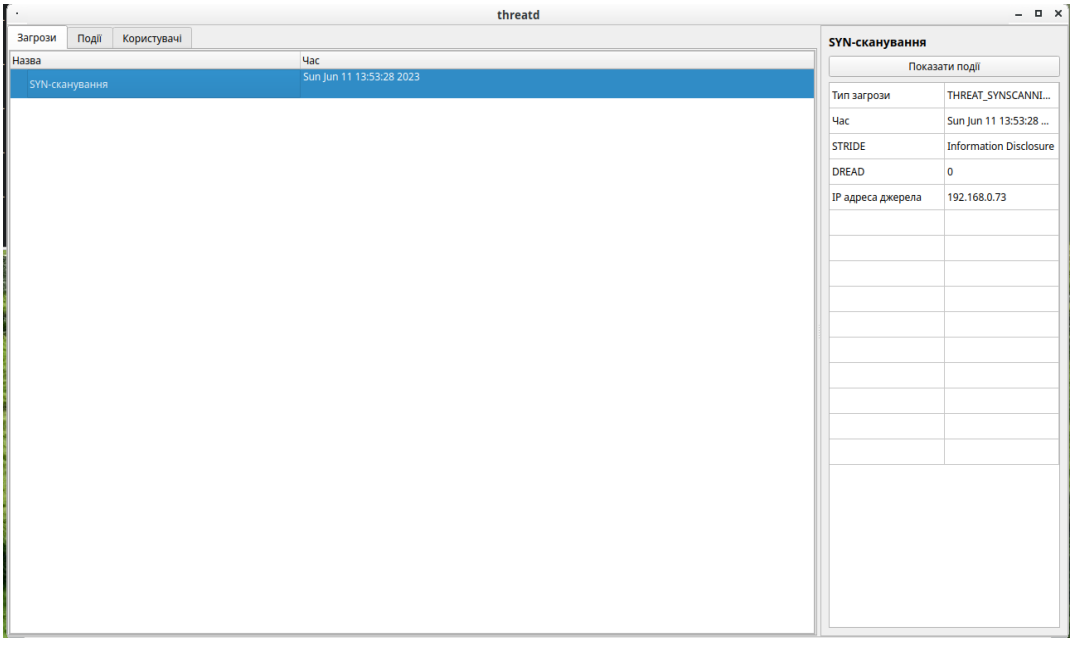

Рисунок 3.12 – Перегляд загроз у вікні програмного засобу

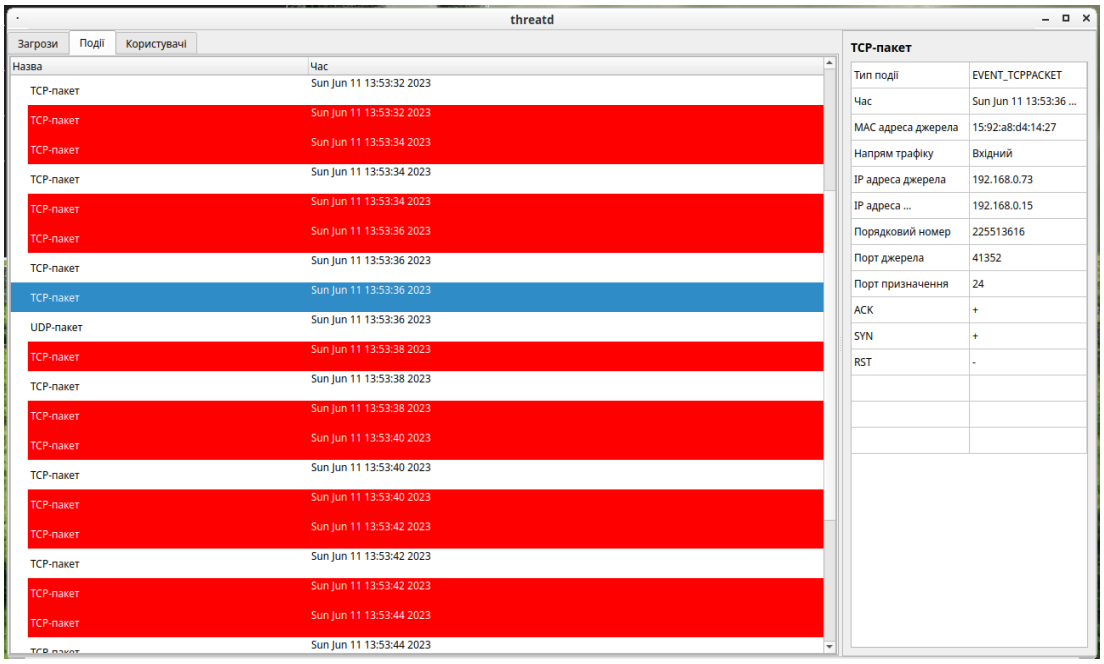

Рисунок 3.13 – Перегляд подій, що становлять загрозу у вікні програмного засобу

Висновки за розділом 3

В даному розділі була розроблена структура програмного засобу аналізу загроз в інформаційній системі. Ця структура складається з кількох компонентів, що взаємодіють між собою. Першим компонентом є компонент збору даних, який відповідає за отримання та збір інформації з різних джерел. Важливим етапом розробки структури є правила виявлення загроз, які дозволяють встановити критерії та алгоритми для виявлення підозрілої або ворожої активності.

Наступним компонентом є компонент виявлення загроз, який використовує правила з попереднього етапу для аналізу та ідентифікації потенційних загроз. Цей компонент виконує обробку зібраних даних та визначає, чи відповідають вони заданим критеріям загроз.

Сховище даних є важливою складовою структури програмного засобу, оскільки воно забезпечує збереження та доступ до оброблених даних. Дані можуть бути збережені в базі даних або в іншій формі, яка найкраще підходить для подальшого аналізу та використання.

Для програмної реалізації даного засобу була обрана мова програмування C++ та ряд бібліотек, що допомагають здійснити аналіз та виявлення загроз. Однією з

таких бібліотек є бібліотека libtins, яка дозволяє працювати з мережевими пакетами та виконувати аналіз їх заголовків. Крім того, для обробки логів та нормалізації журнальних повідомлень може бути використана бібліотека liblognorm.

Був наведений приклад виявлення загрози сканування портів за допомогою розробленого програмного засобу аналізу загроз.

ВИСНОВКИ

У результаті проведеного аналізу загроз кібербезпеки та моделі їх класифікації та пріоритизації, були використані методики STRIDE і DREAD. STRIDE дозволяє класифікувати загрози залежно від їх впливу на систему, а DREAD визначає ризик загрози на основі п'яти факторів. Використання цих методик допомагає систематизувати та оцінити загрози з точки зору їх впливу та ризику, що дозволяє приймати обґрунтовані рішення щодо заходів захисту та пріоритетів у виконанні цих заходів.

Дослідження процесу виявлення загроз показало, що він складається з кількох етапів. Перший етап - це збір даних, який включає моніторинг мережевого трафіку, запис журналів подій, збір інформації про вразливості системи та інші джерела даних. Наступний етап - попередня обробка даних, що включає їх структурування, фільтрацію, перетворення до єдиного формату та видалення шуму або помилок. Останній етап - співставлення даних для виявлення загроз, де застосовуються алгоритми та методи аналізу для виявлення незвичайних патернів, аномалій або вразливостей, що можуть вказувати на наявність загроз.

Розробка програмної реалізації ефективного аналізу та ідентифікації загроз в інформаційній системі є важливим завданням. Це дозволяє автоматизувати процес виявлення загроз, що спрощує роботу адміністраторів та забезпечує більш точний та швидкий аналіз безпекових подій. Реалізація включає в себе розробку алгоритмів та програмних модулів, які враховують методики класифікації та пріоритизації загроз, а також виконують етапи збору, обробки та співставлення даних. Результатом такої реалізації є покращена реакція на загрози, зменшення часу реагування та забезпечення більшої надійності та безпеки інформаційної системи.

Загалом, аналіз загроз, дослідження процесу виявлення загроз та розробка програмної реалізації для ефективного аналізу та ідентифікації загроз є важливими кроками у забезпеченні кібербезпеки. Ці дії дозволяють зрозуміти потенційні загрози, оцінити їх ризик та вплив, а також розробити та впровадити відповідні заходи

захисту. Результатом є збалансований підхід до забезпечення безпеки інформаційних систем, що дозволяє ефективно протистояти сучасним кіберзагрозам і зберігати надійність та конфіденційність даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. NIST Special Publication 800-30 Rev. 1 " Guide for Conducting Risk Assessments”, 2012, 95 p.

2. The STRIDE Threat Model [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)

3. CAPEC-151: Identity Spoofing [Електронний ресурс]. – Режим доступу: https://capec.mitre.org/data/definitions/151.html

4. SSL Spoofing [Електронний ресурс]. – Режим доступу: https://owasp.org/www-pdf-archive/SSL_Spoofing.pdf

5. CAPEC-142: DNS Cache Poisoning [Електронний ресурс]. – Режим доступу: https://capec.mitre.org/data/definitions/142.html

6. Repudiation Attack [Електронний ресурс]. – Режим доступу: https://owasp.org/www-community/attacks/Repudiation_Attack

7. Sensitive Data Exposure [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

8. Unsecured Credentials: Private Keys [Електронний ресурс]. – Режим доступу: https://attack.mitre.org/techniques/T1552/004/

9. Denial of Service [Електронний ресурс]. – Режим доступу: https://attack.mitre.org/techniques/T0814/

10. Traffic flood [Електронний ресурс]. – Режим доступу: https://owasp.org/www-community/attacks/Traffic_flood

11. Account Manipulation [Електронний ресурс]. – Режим доступу: https://attack.mitre.org/techniques/T1098/

12. Adam Shostack. Threat Modeling: Designing for Security. Indianapolis : “Wiley”, 2014, 590 p.

13. DREADful [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/archive/blogs/david_leblanc/dreadful

14. A Complete Guide to the Common Vulnerability Scoring System [Електронний ресурс]. – Режим доступу: https://www.first.org/cvss/v2/guide

15. Understanding the NIST cybersecurity framework [Електронний ресурс]. – Режим доступу: https://www.ftc.gov/business-guidance/small-businesses/cybersecurity/nist-framework

16. Threat Hunting: Data Collection and Analysis [Електронний ресурс]. – Режим доступу: https://resources.infosecinstitute.com/topic/threat-hunting-data-collection-and-analysis/

17. Data Standardization [Електронний ресурс]. – Режим доступу: https://threathunterplaybook.com/pre-hunt/data_standardization.html

18. David R. Miller, Shon Harris, Allen Harper, Stephen VanDyke, Chris Blask. Security Information and Event Management (SIEM) Implementation. New York : “The McGraw-Hill Companies”, 2010, 464 p.

19. MITRE ATT&CK: Data Sources [Електронний ресурс]. – Режим доступу: https://attack.mitre.org/datasources/

20. The Importance of Log Management and Cybersecurity [Електронний ресурс]. – Режим доступу: https://www.graylog.org/post/the-importance-of-log-management-and-cybersecurity/

21. The Importance of Network Traffic Analysis (NTA) for SOCs [Електронний ресурс]. – Режим ресурсу: https://www.cybersecurity-insiders.com/portfolio/the-importance-of-network-traffic-analysis-nta-soc-webinar/

22. signal(7) — Linux manual page [Електронний ресурс]. – Режим доступу: https://man7.org/linux/man-pages/man7/signal.7.html

23. David Clinton. Linux in Action. Shelter Island : “Manning”, 2018, 384 p.

24. NIST Special Publication 800-92 " Guide to Computer Security Log Managemen”, 2006, 72 p.

25. SMTP Server Logs [Електронний ресурс]. – Режим доступу: https://www.informit.com/articles/article.aspx?p=24263&seqNum=7

26. Chris Sanders, Jason Smith .Applied Network Security Monitoring: Collection, Detection, and Analysis. Waltham : “Syngress”, 2014, 496 p

27. How to Integrate Data from Multiple Sources: 5 Challenges to Overcome [Електронний ресурс]. – Режим доступу: https://medium.com/instinctools/how-to-integrate-data-from-multiple-sources-5-challenges-to-overcome-37992c8dfbf5

28. Adversary-in-the-Middle [Електронний ресурс]. – Режим доступу: https://attack.mitre.org/techniques/T1557/

29. Ke Wang, Salvatore J. Stolfo. Anomalous Payload-based Network Intrusion Detection. New York : Columbia University, 20 p.

30. A Firewall Log Analysis Primer [Електронний ресурс]. – https://www.secureworks.com/blog/firewall-primer

31. Andreas Müller. Event Correlation Engine. Zurich : Swiss Federal Institute of Technology Zurich, 2009, 175 p.

32. Serialization and Unserialization [Електронний ресурс]. – Режим доступу: https://web.archive.org/web/20150405013606/http://isocpp.org/wiki/faq/serialization

33. LevelDB Documentation [Електронний ресурс]. – Режим доступу: https://github.com/google/leveldb/blob/main/doc/index.md

34. Examples – libtins [Електронний ресурс]. – Режим доступу: http://libtins.github.io/examples/

35. fanotify(7) — Linux manual page [Електронний ресурс]. – Режим доступу: https://man7.org/linux/man-pages/man7/fanotify.7.html

36. Liblognorm 1.1.2 documentation [Електронний ресурс]. – Режим доступу: https://www.liblognorm.com/files/manual/index.html