НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

До захисту допущено

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інтегровані інформаційні системи»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Система функціональної безпеки технологічного процесу»

Виконав (-ла):
студент (-ка) IV курсу, групи ІА-92
Сус Кирило Матвійович _____

Керівник:
Корнієнко Богдан Ярославович,
професор, д.т.н., професор
_____

Рецензент:
Доцент кафедри ТПЗА ІХФ,
к.т.н., доцент
Ладієва Л. Р. _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 126 «Інформаційні системи та технології»
Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Сусу Кирилу Матвійовичу

1. Тема проєкту «Система функціональної безпеки технологічного процесу», керівник проєкту Корнієнко Богдан Ярославович, професор, д.т.н., професор, затверджені наказом по університету від «31» травня 2023 р. № 2101-с

2. Термін подання студентом проєкту 15 червня 2023 р.

3. Вихідні дані до проєкту: математична модель впливу загроз на інформаційну систему обробки персональних даних, мова програмування Python.

4. Зміст пояснювальної записки: вступ, математична модель системи функціональної безпеки, розробка системи функціональної безпеки для технологічного процесу, структура програми, висновки.

5. Перелік графічного матеріалу: 4 креслення, 18 зображень.

7. Дата видачі завдання 27.02.2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Отримати завдання на ДП.	27.02.23 – 28.02.23	
2	Розділ 1 ДП, ВСТУП, Літературний огляд.	17.04.23 – 24.04.23	
3	Розділ 2 ДП Основна частина.	01.05.23 – 05.05.23	
4	Розділ 3 ДП Практична частина. ВИСНОВКИ.	08.05.23 – 12.05.23	
5	Підготовка супровідної документації	15.05.23 – 19.05.23	
6	Подання роботи керівнику ДП	21.05.23 – 27.05.23	
7	Перевірка роботи на плагіат (Plagiarism Detector)	29.05.23 – 09.06.23	
8	Подання ДП рецензенту. Отримання рецензії.	5.06.23 – 12.06.23	
9	Подання в електронному вигляді ДП та анотації до неї	12.06.23 – 16.06.23	
10	Захист ДП	19.06.23 – 23.06.23	

Студент Кирило СУС

Керівник проєкту Богдан КОРНІЄНКО

АНОТАЦІЯ

Сус К. М. Система функціональної безпеки технологічного процесу. КПІ ім. Ігоря Сікорського, Київ, 2023.

Пояснювальна записка містить: вступ, три розділи, висновки, список використаної літератури із 43 джерел. Загальний обсяг дипломного проєкту складає: 71, основного тексту – 60, рисунків – 18 та 4 конструкторських документів.

Об’єктом дослідження є матиматична модель система функціональної безпеки.

Мета розробки – дослідити матиматичну модель система функціональної безпеки .

У дипломному проєкті була розроблена система з використанням мови програмування Python. Ця мова програмування відома своєю зрозумілістю та гнучкістю. Результатом аналізів є спроектована модель системи функціональної безпеки.

Отримані результати можуть бути корисними при досліджені аналогічних чи подібних об’єктів, а також для покращення функціональності та ефективності системи безпеки технологічного процесу.

SUMMARY

Sus K. M. System of functional safety of the technological process.

KPI named after Igor Sikorskyi, Kyiv, 2023.

The explanatory note contains: an introduction, three chapters, conclusions, a list of used literature from 43 sources. The total volume of the diploma project is: 71, main text – 60, drawings – 19 and 4 design documents.

The object of the study is a mathematical model of the functional safety system.

The purpose of the development is to investigate the mathematical model of the functional safety system.

In the diploma project, a system was developed using the Python programming language. This programming language is known for its simplicity and flexibility. The result of the analysis is the designed model of the functional safety system.

The obtained results can be useful in the study of similar or similar objects, as well as for improving the functionality and efficiency of the safety system of the technological process.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА92.220БАК.004 ПЗ	Пояснювальна записка	60		
6	А3	ІА92.220БАК.004 Д1	Діаграма взаємозв’язків	1		
7	А3	ІА92.220БАК.004 Д2	Діаграма классів	1		
8	А3	ІА92.220БАК.004 Д3	Діаграма архітектура системи	1		
9	А3	ІА92.220БАК.004 Д4	Діаграма випадків	1		
10			використання			
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІА92.220БАК.004 ТП			
Розроб.		Сус К. М.			Застосунок з автоматичною генерацією звітності для благодійних організацій. Відомість проєкту	Літ.	Аркуш	Аркушів
Керівн.		Корнієнко Б. Я.				Т	1	1
						КПІ ім. Ігоря Сікорського		
Затв.								

Пояснювальна записка
до дипломного проєкту
на тему: «Система функціональної безпеки технологічного процесу»

Київ – 2023 року

ЗМІСТ

ВСТУП

У сучасному світі, де технологічні процеси займають важливе місце у багатьох сферах діяльності, питання забезпечення безпеки цих процесів набуває особливої ваги. Система функціональної безпеки технологічного процесу є одним з основних інструментів для забезпечення безпеки в цьому контексті.

Мета дослідження полягає в розробці та впровадженні ефективної системи функціональної безпеки для технологічного процесу з метою запобігання можливим аварійним ситуаціям, збереження ресурсів та забезпечення надійності функціонування.

Актуальність дослідження обумовлена необхідністю забезпечення безпеки в технологічних процесах, особливо в галузях, де відбуваються складні та потенційно небезпечні операції. Недостатня безпека може призвести до серйозних наслідків, які включають матеріальні збитки, негативний вплив на довкілля та загрозу життю людей. Розробка системи функціональної безпеки відповідає потребам сучасного ринку та сприяє забезпеченню стабільності технологічних процесів.

Об'єктом дослідження є технологічний процес, що включає в себе послідовність операцій, які проводяться з метою отримання певного продукту або послуги.

Предметом дослідження є система функціональної безпеки, що включає в себе методи, алгоритми та інструменти, спрямовані на забезпечення безпеки технологічного процесу шляхом виявлення, аналізу та реагування на потенційні загрози та аварійні ситуації.

Дослідження системи функціональної безпеки технологічного процесу має велике значення для підвищення ефективності, надійності та безпеки в промислових, енергетичних, транспортних та інших галузях, де використовуються складні технологічні системи.

Результати дослідження можуть бути використані для розробки та впровадження нових підходів до забезпечення безпеки технологічних процесів, що сприятиме стабільному розвитку та забезпеченню безпеки відповідних галузей.

1 ОГЛЯД ЛІТЕРАТУРИ

1.1 Поняття функціональної безпеки

Функціональна безпека технічного процесу визначається як система організаційних заходів і технічних засобів, спрямованих на запобігання впливу небезпечних виробничих факторів на працівників.

Технічна безпека виробничих процесів включає в себе безпеку обладнання, що досягається шляхом врахування вимог безпеки на кожному етапі його створення - від складання технічного завдання та проектування до випробувань та передачі його в серійне виробництво. Для забезпечення безпеки виробничих процесів також встановлені загальні вимоги згідно зі стандартами.

Безпека виробничого обладнання досягається за допомогою таких заходів:

- Вибір безпечних принципів дії, конструктивних рішень та елементів.
- Використання засобів механізації, автоматизації та дистанційного керування.
- Застосування засобів захисту в конструкції.
- Дотримання ергономічних вимог.
- Включення вимог безпеки в технічну документацію, пов'язану з монтажем, експлуатацією, ремонтом та транспортуванням обладнання.
- Використання відповідних матеріалів у конструкції обладнання.
- Вимоги безпеки повинні бути враховані на всіх етапах проектування, тому вони включені у всі види проектної документації, такі як технічні завдання, технічні умови та стандарти на виробниче обладнання. При виборі принципу дії машини необхідно враховувати всі потенційно небезпечні виробничі чинники.

Це означає, що воно не повинно створювати небезпечних ситуацій через вплив різних факторів, таких як вологість, сонячна радіація, механічні коливання, високий і низький тиск, агресивні речовини, мікроорганізми, грибки, комахи тощо. Крім того, обладнання не повинно забруднювати довкілля шкідливими речовинами понад нормативні значення.

Виробниче обладнання повинно відповідати вимогам безпеки на всіх етапах його життєвого циклу, включаючи монтаж, експлуатацію, ремонт, транспортування та зберігання. Нові речовини і матеріали мають пройти відповідну гігієнічну перевірку та випробування на пожежну безпеку, перш ніж їх можна використовувати.

Дистанційне керування забезпечує контроль та регулювання роботи обладнання з безпечних відстаней від небезпечних зон.

У процесі виготовлення машин і обладнання необхідно враховувати ергономічні вимоги. Це означає, що вони повинні бути відповідними антропометричним особливостям людини, враховувати психофізіологічні аспекти та задовольняти естетичні вимоги.

Таким чином, функціональна безпека технічного процесу включає в себе систему заходів і засобів, які спрямовані на запобігання впливу небезпечних факторів на працівників та забезпечення безпечної експлуатації обладнання на всіх етапах його життєвого циклу. Дотримання вимог безпеки є важливим аспектом у всіх стадіях проектування технічного процесу та виробництва обладнання.

1.2 Безпека технологічного процесу

Вимоги до виробничих процесів, передбачають наступне:

- Уникнення прямого контакту працівників з небезпечними матеріалами, заготовками, напівфабрикатами, готовою продукцією та відходами виробництва.

- Заміна технологічних процесів та операцій, що породжують небезпечні та шкідливі виробничі фактори, на безпечні або менш інтенсивні.
- Використання комплексної механізації та автоматизації виробництва.
- Застосування дистанційного керування технологічними процесами та операціями в разі наявності небезпечних і шкідливих виробничих факторів.
- Герметизація обладнання.
- Використання колективних засобів захисту працівників.
- Раціональна організація праці та відпочинку для запобігання монотонності та гіподинамії, а також зменшення напруженості праці.
- Своєчасне отримання інформації про виникнення небезпечних та шкідливих виробничих факторів на окремих технологічних операціях.
- Впровадження систем керування технологічними процесами, які забезпечують захист працівників та аварійне вимкнення виробничого обладнання.
- Своєчасне видалення та безпечне утилізація відходів виробництва, що становлять джерела небезпечних та шкідливих виробничих факторів.
- Забезпечення пожежної та вибухобезпеки.

Розташування виробничого обладнання, матеріалів, заготовок, напівфабрикатів, готової продукції та відходів виробництва в приміщеннях та на робочих місцях не повинно становити загрози для персоналу. Розміщення обладнання та комунікацій, що породжують небезпечні та шкідливі виробничі фактори, а також відстань між одиницями обладнання та між обладнанням і стінами будівель повинні відповідати вимогам діючих норм технологічного проектування та будівельних правил.

1.3 Вимоги безпеки щодо організації робочих місць

Організація робочого місця повинна враховувати фізіологічні, антропометричні та психофізіологічні особливості працівників, а також характер виконуваної роботи. Робоче місце повинно бути облаштоване згідно з вимогами стандартів, що забезпечує комфортне положення людини. Це може бути досягнуто шляхом регулювання положення крісла, висоти та кута нахилу підставки для ніг або розмірів робочої поверхні.

Загальні принципи організації робочого місця включають:

- Видалення зайвих предметів з робочого місця, збереження необхідних предметів поруч, але не заважаючи працівникові.
- Розміщення часто використовуваних предметів ближче до працівника, ніж рідко використовувані.
- Правильне розташування предметів відповідно до використовуваних рук: ліворуч для предметів, що використовуються лівою рукою, та праворуч для предметів, що використовуються правою рукою.
- Якщо для роботи використовуються обидві руки, місце розташування пристосувань повинно бути зручним для захоплення обома руками.
- Небезпечне обладнання має бути розташоване вище, ніж менш небезпечне, з метою зменшення ризику травмування працівника. Проте, слід враховувати, що важкі предмети краще опускати, аніж піднімати.
- Робоче місце не повинно бути захаращене заготовками або готовими деталями.
- Важливо забезпечити достатню оглядовість на робочому місці.

Засоби відображення інформації мають бути розташовані в зонах інформаційного поля робочого місця, враховуючи частоту та значущість інформації, тип засобів відображення, швидкість та точність сприйняття і зчитування.

Засоби захисту широко використовуються на виробництві з метою запобігання або зменшення впливу небезпечних і шкідливих виробничих факторів на працюючих. Ці засоби поділяються на колективний та індивідуальний захист.

Колективний захист - це засоби, які призначені для захисту одночасно двох або більше працюючих. Вони включають огороджувальні та запобіжні улаштування, блокування, сигналізатори небезпеки, розпізнавальне фарбування і знаки небезпеки, дистанційне керування, а також спеціальні засоби безпеки, які можуть бути частинами технічного обладнання або використовуватися виробничих приміщеннях.

Огородження призначені для ізоляції небезпечних зон та запобігання доступу до них працівникам. Залежно від способу встановлення і особливостей експлуатації, огородження поділяються на знімні, відкидні, розсувні та відкриваються. Знімні огородження та огородження, що відкриваються, застосовуються тільки у випадках, коли неможливо встановити незнімні огородження залежно від конструктивних особливостей обладнання. Ці огородження повинні бути заблоковані з пусковим, сигнальним або гальмівним пристроєм обладнання.

Запобіжні засоби призначені для автоматичного вимкнення машин і механізмів, якщо їх робочі параметри або виробничі умови відхиляються від допустимих значень. Застосовуються такі запобіжні засоби, як електроконтактні термометри, запобіжні клапани, гальмівні улаштування, кінцеві вимикачі та реле захисту від великих струмів.

Блокування поділяються на механічні, електричні, фотоелектричні, пневматичні, гідравлічні та комбіновані. Вони перешкоджають функціонуванню обладнання без наявності засобів безпеки.

Сигналізатори про небезпеку інформують про роботу обладнання та наявність небезпечних і шкідливих факторів. Їх поділяють на звукові, візуальні, комбіновані (світлозвукові) та одоризаційні.

Сигналізація може бути оперативною, попереджувальною або розпізнавальною. Використовуються знаки, які поділяються на заборонні, попереджувальні, приписуючі та вказівні в залежності від їхньої цільової функції.

1.4 Математичні моделі систем безпеки

Формальне визначення політики безпеки є математичною моделлю безпеки, яка використовується у галузі захисту інформації в інформаційних системах. Ці системи будуються на основі математичних моделей, що дозволяє теоретично обґрунтувати відповідність системи захисту інформації вимогам політики безпеки.

Розвиток формальної теорії захисту інформації призначений для побудови сучасних систем захисту інформації, існує багато математичних моделей, що описують різні аспекти безпеки та надають теоретичну базу для цих систем. Деякі поширені моделі безпеки включають Харрісона-Руззо-Ульмана, Take-Grant, Белла-ЛаПадула та інші.

Для захисту від апаратних та програмних закладок, які можуть бути впроваджені на етапах розробки та виробництва, необхідно приймати відповідні заходи. Несанкціонована зміна структури комп'ютерних систем, яка може відбутися на етапах розробки або експлуатації, називається закладками.

На різних етапах життєвого циклу комп'ютерних систем вирішуються різні задачі щодо захисту від загроз. Під час розробки та модернізації системи важливо максимально уникнути помилок та закладок, а під час експлуатації необхідно контролювати та запобігати помилкам та закладкам, забезпечувати цілісність та незмінність структур системи.

Для запобігання несанкціонованій зміні проектних рішень на рівнях алгоритмічної, програмної або технічної структури комп'ютерних систем слід дотримуватись таких основних принципів:

- Залучення висококваліфікованих фахівців до розробки.

- Використання ієрархічних структур, стандартних конструкцій та блоків, сучасних технологій програмування під час розробки.
- Автоматизація процесу розробки.
- Ретельний контроль процесу розробки.
- Сертифікація кінцевого продукту та інші заходи.
- З метою захисту комп'ютерної системи від несанкціонованої зміни її структури під час експлуатації, необхідно вживати наступні заходи:
- Забезпечення охорони приміщень.
- Розмежування доступу до обладнання.
- Протидія несанкціонованому підключенню обладнання.
- Захист засобів управління, комутації та внутрішнього монтажу від несанкціонованого втручання.
- Протидія впровадженню шкідливих програм.

Захист від несанкціонованого доступу до інформації у комп'ютерних системах є найнебезпечнішою загрозою для безпеки інформації. Несанкціонований доступ зазвичай здійснюється за допомогою знань про комп'ютерну систему та навичок роботи з нею, інформації про систему захисту, вразливостей програмного та апаратного забезпечення, а також через помилки та необережність обслуговуючого персоналу та користувачів.

Для захисту інформації від несанкціонованого доступу використовується система розмежування доступу (СРД). СРД є ефективним механізмом комплексного захисту інформації. Однак, зловмисник може досліджувати СРД з метою несанкціонованого доступу шляхом дослідження та копіювання інформації про систему захисту (СЗІ). Для запобігання цій загрозі використовується комплекс засобів і заходів, створюється система захисту від дослідження і копіювання інформації (СЗДК). Таким чином, СРД та СЗДК є підсистемами системи захисту інформації від несанкціонованого доступу.

СРД базується на певній моделі політики безпеки, при чому найпоширеніші моделі цього типу - дискреційна та мандатна політики безпеки.

Основними елементами СРД є блок ідентифікації та автентифікації суб'єктів доступу, диспетчер доступу, блок криптографічного перетворення інформації при її збереженні та передачі, а також блок очищення пам'яті.

До того ж, математичні моделі систем безпеки є потужним інструментом для аналізу, проектування та оптимізації безпекових систем. Вони дозволяють відтворити складні взаємодії та поведінку системи, що допомагає зрозуміти й передбачити її реакцію на різні впливи та загрози.

Основні типи математичних моделей включають:

- Імовірнісні моделі: Ці моделі використовують теорію ймовірностей для опису імовірностей виникнення подій та ступеня ризику. Вони базуються на аналізі статистичних даних, історичних звітів та інших джерел інформації. Імовірнісні моделі дозволяють оцінити ймовірність виникнення певної загрози або аварійної ситуації, що допомагає приймати обґрунтовані рішення щодо запобігання цим подіям та управління ризиками.
- Статичні моделі: Ці моделі базуються на математичних методах аналізу та моделювання статичних структур систем безпеки. Вони дозволяють описати структуру системи, виявити слабкі місця та потенційні загрози. Статичні моделі часто використовуються для аналізу архітектури мереж, систем контролю доступу, систем виявлення вторгнень та інших компонентів безпекових систем.
- Динамічні моделі: Ці моделі описують зміну стану системи з часом та її реакцію на зовнішні впливи. Вони використовуються для моделювання процесів безпеки, динаміки виявлення та реагування на загрози,

- прогнозування наслідків аварійних ситуацій та оцінки ефективності заходів безпеки. Динамічні моделі можуть бути представлені у вигляді різних математичних формалізмів, таких як диференціальні рівняння, автомати, сітки Петрі та інші.
- Оптимізаційні моделі: Ці моделі використовуються для знаходження оптимальних рішень та розподілу ресурсів у системах безпеки. Вони зазвичай базуються на оптимізаційних алгоритмах та методах, що дозволяють знайти найкращі рішення з урахуванням обмежень та цілей безпекової системи.

Математичні моделі систем безпеки допомагають зрозуміти складність взаємодій та залежностей в безпекових системах, а також сприяють розробці та вдосконаленню ефективних стратегій безпеки. Вони є необхідним інструментом для аналізу та управління ризиками в технологічних процесах та сприяють підвищенню рівня безпеки в різних галузях діяльності.

1.5 Аналіз існуючих систем функціональної безпеки

Аналіз існуючих систем функціональної безпеки може охоплювати різні аспекти та аспекти систем. Однак, основним завданням систем функціональної безпеки є забезпечення безпеки функціонального функціонування системи та захист від несанкціонованого доступу, помилок та інших загроз, що можуть впливати на її нормальну роботу.

Однією з найбільш відомих систем функціональної безпеки є Safety Instrumented Systems (SIS) - системи безпечного керування. Вони використовуються в промислових об'єктах, таких як хімічні заводи, нафтопереробні підприємства тощо, для контролю за процесами та захисту від аварійних ситуацій. SIS включають датчики, логічні контролери та виконавчі пристрої, які вживають заходів для запобігання небезпечним ситуаціям або зниження їх наслідків.

Також варто згадати систему функціональної безпеки в інформаційних технологіях, де забезпечення безпеки програмного забезпечення та систем є важливим аспектом. Такі системи включають контроль доступу, шифрування, перевірку цілісності даних та інші заходи для захисту від несанкціонованого доступу та кібератак.

Нижче наведена таблиця, в якій проаналізовані деякі існуючі системи функціональної безпеки. Аналіз здійснюється на основі різних критеріїв, включаючи архітектуру, можливі загрози, уразливості та ефективність заходів безпеки.

Таблиця 1.1. – Аналіз систем функціональної безпеки

Система безпеки	**Архітектура**	**Загрози**	**Уразливості**	**Ефективність заходів безпеки**
Система A	Централізована	Зломи, вторгнення	Відсутність шифрування, слабкі паролі	Висока
Система B	Децентралізована	Віруси, витоки інформації	Несанкціонований доступ до файлів	Середня
Система C	Розподілена	Соціальний інжиніринг, фішинг	Вразливості в мережевих протоколах	Низька
Система D	Ієрархічна	Витоки конфіденційної інформації	Недостатня аутентифікація та авторизація	Висока
Система E	Гібридна	Зломи, вторгнення, соціальний інжиніринг	Вразливості веб-додатків	Середня

Таблиця надає огляд різних систем функціональної безпеки, їх архітектури, основних загроз та уразливостей. Також вказано ефективність заходів безпеки для кожної системи. Варто зазначити, що це лише приклади систем і аналіз може бути більш глибоким та детальним в конкретному дослідженні.

Висновки до розділу

У ході огляду літератури функціональної безпеки було проведено аналіз різних аспектів та систем, пов'язаних з захистом функціонального функціонування систем від потенційних загроз. Було розглянуто різні типи систем функціональної безпеки, такі як Safety Instrumented Systems (SIS) для промислових об'єктів.

Розділ підкреслює важливість розуміння та впровадження систем функціональної безпеки для забезпечення безпеки функціонування систем у різних галузях. Подальше дослідження та розвиток в цій області можуть сприяти покращенню захисту від потенційних загроз і забезпеченню безпеки та надійності систем

В цілому, огляд літератури надав цінну інформацію про різноманітні системи та підходи до захисту функціонального функціонування систем. Цей огляд слугує базою для подальшого дослідження та розвитку в галузі функціональної безпеки, зокрема для виявлення нових методів і рішень у сфері захисту систем від потенційних загроз.

2 МАТЕМАТИЧНА МОДЕЛЬ СИСТЕМИ ФУНКЦІОНАЛЬНОЇ БЕЗПЕКИ

2.1 Опис шести станів системи безпеки

2.1.1 Не існує загрози

Стан "Не існує загрози"(рисунок 2.1) є одним з основних станів системи безпеки. В цьому стані відсутні будь-які ідентифіковані загрози або потенційні ризики, які можуть спричинити порушення безпеки системи.

Опис цього стану включає такі характеристики:

1) Відсутність виявлених загроз: У цьому стані система безпеки не зафіксована жодна загроза або відхилення, що можуть вплинути на безпеку системи. Всі складові системи функціонують згідно з очікуваннями і не потребують негайних заходів забезпечення безпеки.

2) Відсутність вразливостей: У системи відсутні вразливості, які можуть бути використані зловмисниками для здійснення атаки або порушення безпеки.

3) Ефективне функціонування заходів безпеки: У цьому стані система безпеки успішно виконує всі необхідні заходи безпеки, які були розроблені та реалізовані для запобігання загрозам. Всі механізми безпеки працюють належним чином і забезпечують високий рівень захисту.

4) Низький ризик порушення безпеки: У стані "Не існує загрози" ризик порушення безпеки системи є дуже низьким або мінімальним. Це означає, що імовірність виникнення вразливостей, атак або інших загроз є малою, і система може функціонувати без помітних ризиків для безпеки даних і ресурсів.

5) Задоволення вимог безпеки: У цьому стані система безпеки успішно відповідає всім вимогам безпеки, встановленим для даної системи.

6) Стабільна працездатність: У стані "Не існує загрози" система безпеки функціонує стабільно і не виявляє жодних проблем або відхилень у процесі роботи.

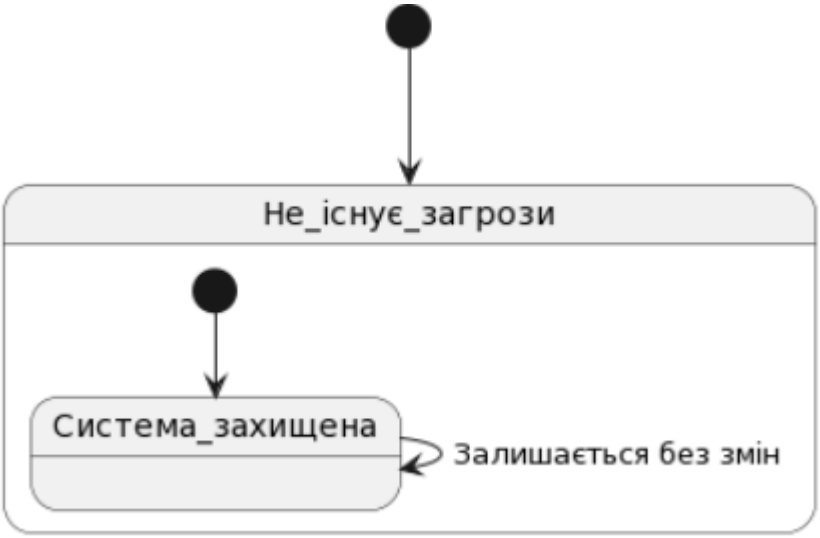

Рисунок 2.1 – Не існує загрози

Цей стан є бажаним для будь-якої системи безпеки, оскільки він вказує на відсутність загроз і ризиків, забезпечує надійність і конфіденційність даних, а також ефективну роботу системи безпеки в цілому. Однак, важливо пам'ятати, що стан "Не існує загрози" може бути тимчасовим, і система повинна постійно моніторитись та оновлюватись для забезпечення стійкості та безпеки у майбутньому.

2.1.2 Загроза існує, але не активна

Стан "Загроза існує, але не активна"(рисунок 2.2) є одним з основних станів системи безпеки. В цьому стані існує визнана загроза безпеці системи, але на даний момент ця загроза не проявляється або не виявляється в активних атаках чи порушеннях.

Опис цього стану включає такі характеристики:

1) Визнана загроза: У цьому стані існує конкретна загроза, яка була виявлена і визнана системою безпеки. Це може бути відома уразливість, потенційна атака або інше можливе порушення безпеки, яке може стати реальним у майбутньому.

2) Відсутність активних атак: Незважаючи на існування загрози, на даний момент не спостерігається активних атак або порушень безпеки. Це може бути результатом належної роботи системи безпеки, запобіжних заходів або відсутності зловмисників, які намагаються експлуатувати загрозу.

3) Стратегії захисту: У цьому стані система безпеки використовує стратегії та заходи для запобігання або зменшення ризику, пов'язаного з існуючою загрозою. Це можуть бути регулярні оновлення програмного забезпечення, контроль доступу, моніторинг або інші заходи, спрямовані на забезпечення безпеки системи.

4) Проактивність: У стані "Загроза існує, але не активна" система безпеки діє проактивно, антиципуючи можливі атаки або порушення безпеки, пов'язані з існуючою загрозою. Вона може використовувати інтелектуальні системи аналізу, прогнозування та виявлення аномалій для виявлення підозрілих дій або вразливостей та попередження про можливі ризики.

5) Стан постійного спостереження: У цьому стані система безпеки знаходиться в постійному спостереженні і виконує моніторинг загрози, оновлюється щодо нових відомостей про загрозу і аналізується її потенційний вплив на систему. Вона готова вжити відповідних заходів безпеки, якщо загроза стане активною.

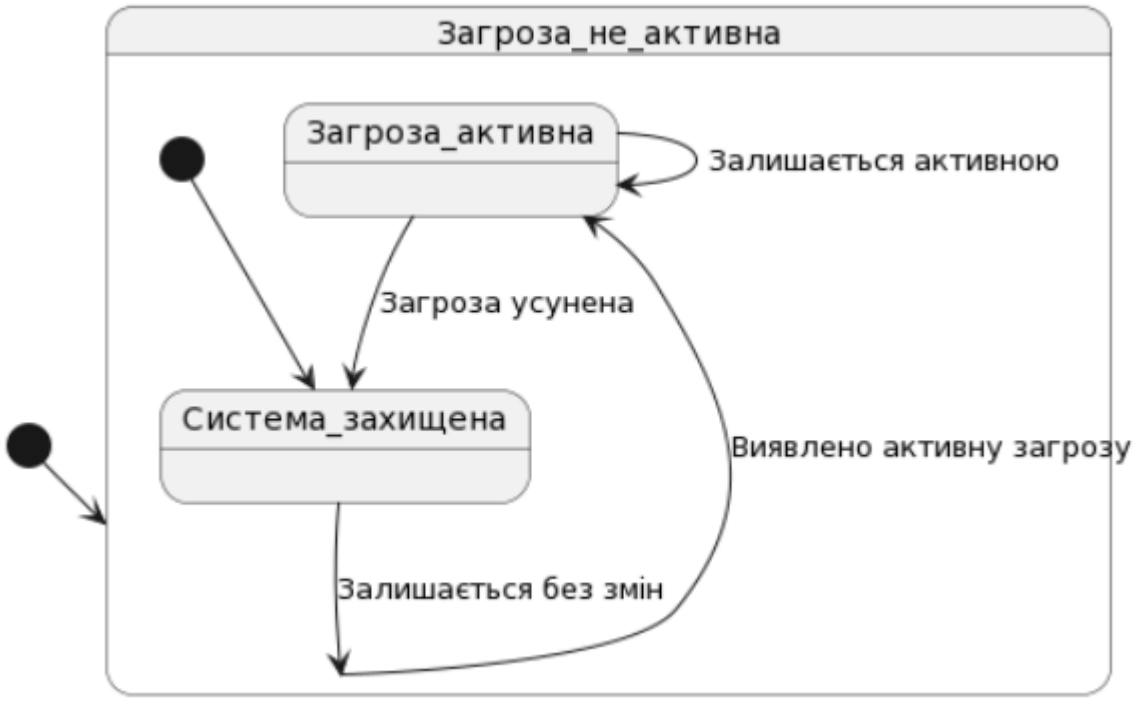

Рисунок 2.2 – Загроза існує, але не активна

Цей стан вказує на необхідність постійного нагляду та реагування на потенційні загрози, навіть якщо вони наразі не виявлені в активних атаках. Система безпеки повинна залишатися підготовленою і вживати заходів для мінімізації ризиків та забезпечення безпеки інформації та ресурсів системи.

2.1.3 Загроза існує і реалізована.

Стан "Загроза існує і реалізована"(рисунок 2.3) є одним з основних станів системи безпеки. У цьому стані загроза, яка була розпізнана раніше, стала реальною і була успішно виконана зловмисниками або іншими небажаними суб'єктами.

Опис цього стану включає такі характеристики:

1) Активна атака: В цьому стані загроза була активно використана для здійснення атаки або порушення безпеки системи. Це може включати несанкціонований доступ до конфіденційної інформації, пошкодження або втрату даних, використання вразливостей системи для впровадження шкідливих програм або інших дій, що можуть негативно вплинути на безпеку системи.

2) Уразливість використана: Зловмисники успішно використали вразливість або слабке місце в системі для реалізації загрози. Це може бути через недостатній рівень захисту, помилки в конфігурації або програмному забезпеченні, відсутність оновлень або інші фактори, які зробили систему уразливою перед атакою.

3) Негативні наслідки: Реалізація загрози має негативні наслідки для системи та її користувачів. Це може включати втрату даних, порушення конфіденційності, пошкодження репутації, фінансові втрати або інші шкоди, які можуть бути завдані системі або її власникам.

4) Реагування на інцидент: При реалізації загрози система безпеки повинна вжити негайних заходів для реагування на інцидент. Це включає виявлення та зупинку атаки, відновлення системи до нормального стану, виявлення вразливостей та їх усунення, аналіз причин інциденту та вжиття запобіжних заходів для майбутнього.

5) Відновлення та укріплення: Після реалізації загрози система безпеки повинна пройти процес відновлення та укріплення. Це включає відновлення пошкоджених даних та ресурсів, аналіз причин інциденту та впровадження виправлень, підвищення рівня захисту та запобігання подібним інцидентам у майбутньому.

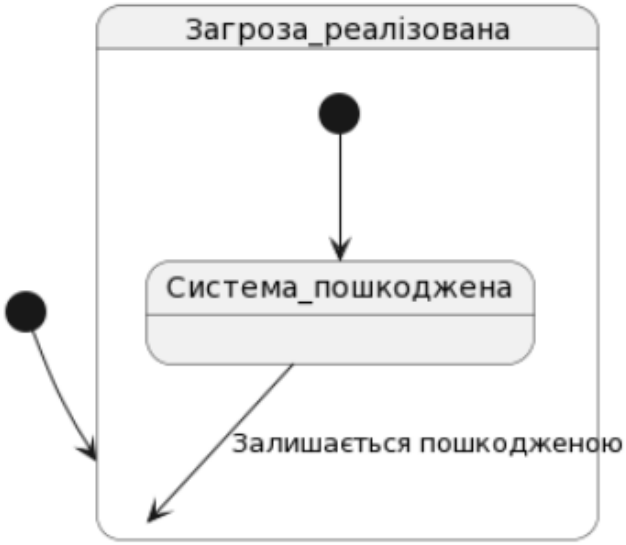

Рисунок 2.3 – Загроза існує, але не активна

Стан "Загроза існує і реалізована" є критичним для системи безпеки, оскільки він означає успішну атаку та можливі наслідки для системи та її користувачів. Відповідне реагування на інцидент, відновлення та укріплення є необхідними кроками для відновлення безпеки системи та запобігання подібним інцидентам у майбутньому.

2.1.4 Загроза існує, реалізована і успішно відбита

Стан "Загроза існує, реалізована і успішно відбита" є одним з основних станів системи безпеки. В цьому стані загроза, яка була розпізнана і активована зловмисниками або іншими небажаними суб'єктами, була ефективно відбита і не завдала шкоди системі або її користувачам.

Опис цього стану включає такі характеристики:

1) Виявлення загрози: Загроза була виявлена і розпізнана системою безпеки. Це може бути завдяки моніторингу системи, виявленню аномальної активності, аналізу журналів подій або використанню спеціалізованих систем виявлення інтрузій.

2) Реалізація загрози: Загроза була активована і почала впливати на систему або її компоненти. Зловмисники можуть намагатись отримати несанкціонований доступ, використовувати вразливості, запускати шкідливі програми або здійснювати інші дії, що можуть призвести до компрометації безпеки.

3) Ефективна реакція: Система безпеки успішно відреагувала на загрозу і вжила відповідних заходів для її відбиття. Це може включати блокування доступу, виявлення інтрузії, використання захисних механізмів або автоматичне відновлення системи.

4) Завершення загрози: Загроза була повністю припинена і зловмисники не змогли завдати шкоди системі або її користувачам. Це означає, що атака була ефективно відбита, і система безпеки повернулася до нормального функціонування.

5) Оцінка та підсилення захисту: Після успішного відбиття загрози, система безпеки проводить оцінку і аналіз інциденту. Це допомагає виявити вразливості, слабкі місця і покращити заходи безпеки, щоб запобігти подібним загрозам у майбутньому.

Рисунок 2.4 – Загроза існує, реалізована і успішно відбита

Стан "Загроза існує, реалізована і успішно відбита" показує, що система безпеки ефективно протистояла потенційній загрозі і забезпечила безпеку системи та її користувачів.

2.1.5 Існує дві загрози, перша реалізована, друга не активна

Стан "Існує дві загрози, перша реалізована, друга не активна"(рис 2.5.) відображає ситуацію, коли система безпеки стикається з двома загрозами, причому перша загроза вже була реалізована, тобто її вплив відчутний, а друга загроза поки що не активна, але є потенційною.

Опис цього стану включає такі аспекти:

1) Реалізована перша загроза: Перша загроза була активована і вже впливає на систему або її компоненти. Це може включати атаки, використання вразливостей, шкідливі програми або інші дії, які можуть завдати шкоди безпеці системи.

2) Друга загроза не активна: Друга загроза поки що не проявляється і не впливає на систему. Це означає, що потенційна загроза існує, але її вплив ще не виявлений або неактивний.

3) Виявлення загроз: Система безпеки успішно виявила першу загрозу, що була реалізована. Це може бути результатом моніторингу, аналізу журналів подій або використання систем виявлення інтрузій.

4) Потенційна загроза: Друга загроза була ідентифікована, але її вплив поки що не виявлений. Це може включати виявлення підозрілої активності, вразливостей або інших ознак потенційного вторгнення.

5) Застосування заходів: У відповідь на реалізовану першу загрозу, система безпеки приймає відповідні заходи для її ліквідації або мінімізації шкоди. Водночас, система може вжити попереджувальних заходів щодо другої загрози, навіть якщо вона поки що не активна.

6) Моніторинг і аналіз: В цьому стані система безпеки здійснює постійний моніторинг та аналіз ситуації з метою виявлення активності або активації другої загрози. Це може включати моніторинг мережі, перевірку системних журналів, використання систем виявлення інтрузій та інших методів аналізу.

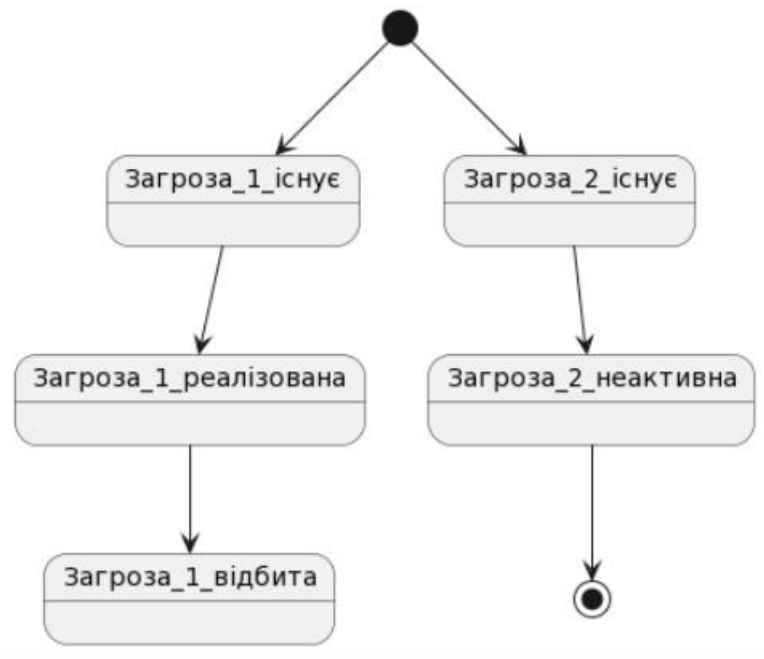

Рисунок 2.5 – Загроза існує, реалізована і успішно відбита

В цілому, цей стан вказує на те, що система безпеки знаходиться в активному режимі реагування на першу реалізовану загрозу, водночас беручи заходи для запобігання або готування до можливої активності другої загрози.

2.1.6 Існує дві загрози обидві реалізовані

Стан "Існує дві загрози, обидві реалізовані" відображає ситуацію, коли система безпеки стикається з двома загрозами, які вже були реалізовані і впливають на систему одночасно. Цей стан може бути особливо складним і потребує негайної уваги та вжиття заходів для ліквідації загроз і мінімізації їхнього впливу.

Опис цього стану включає такі аспекти:

1) Реалізовані загрози: Обидві загрози були активовані і впливають на систему або її компоненти. Це можуть бути різні види атак, використання вразливостей, шкідливі програми або інші дії, які призводять до порушення безпеки системи

2) Потенційні наслідки: Реалізовані загрози можуть мати різні наслідки для системи. Це може включати втрату конфіденційності, цілісності або доступності даних, порушення нормального функціонування системи, втрату ресурсів або недоступність послуг.

3) Управління інцидентами: В такому стані система безпеки повинна негайно реагувати на обидві реалізовані загрози і вживати заходів для контролю, ліквідації і відновлення нормального стану. Це може включати активізацію інцидентного менеджменту, розробку і реалізацію планів дій, співпрацю зі спеціалізованими службами безпеки та відновлення системи.

4) Аналіз причин: Важливим аспектом управління цим станом є аналіз причин, які призвели до реалізації обох загроз. Це допоможе виявити вразливості в системі, недоліки в політиках безпеки або процесах, а також ідентифікувати можливих зловмисників або джерела загроз.

5) Запобігання майбутнім загрозам: Після ліквідації поточних загроз, система безпеки повинна приділити увагу запобіганню майбутнім подібним загрозам. Це може включати оновлення політик безпеки, підвищення рівня свідомості користувачів, впровадження нових захисних механізмів або покращення процесів виявлення і реагування на загрози.

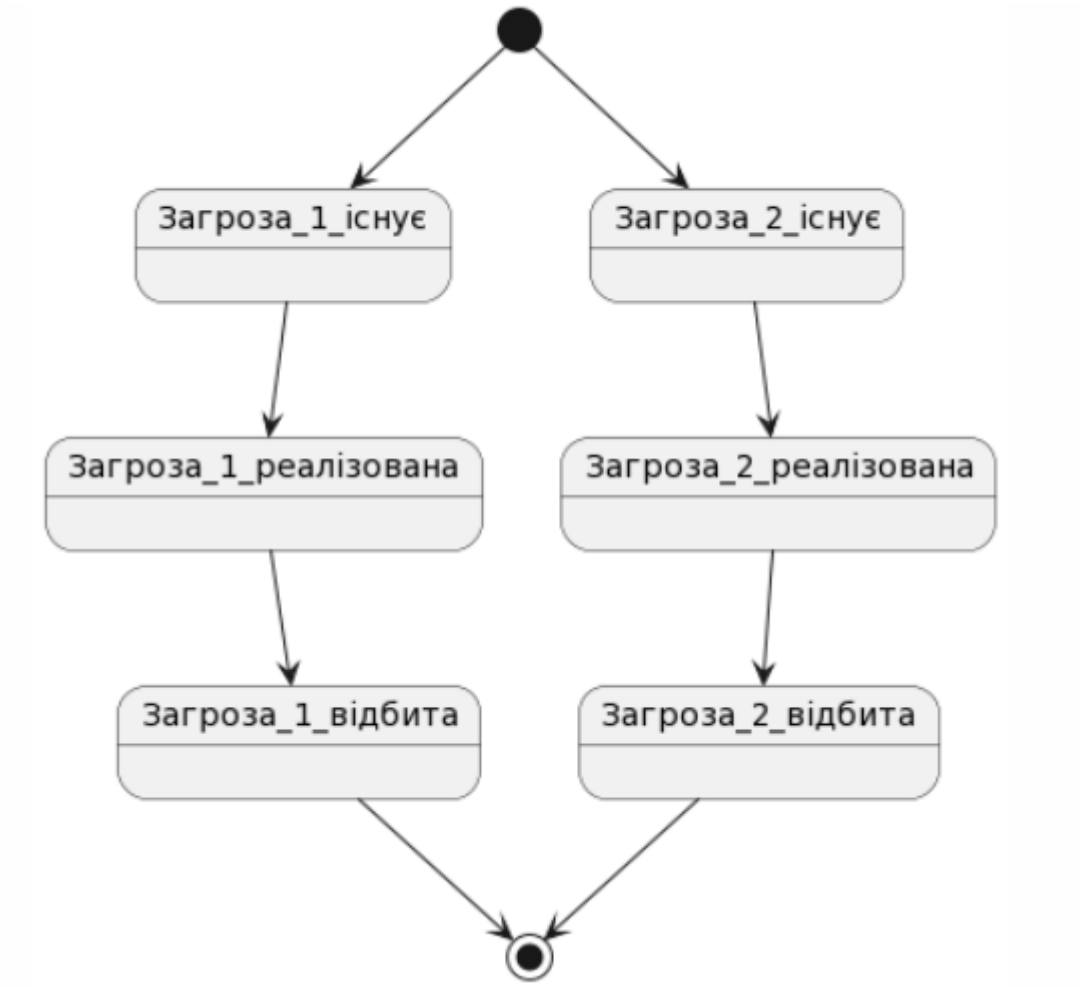

Рисунок 2.6 – Загроза існує, реалізована і успішно відбита

Усі ці аспекти спрямовані на забезпечення безпеки системи після реалізації двох загроз і мінімізацію можливих наслідків.

Відмінність між різними станами системи безпеки полягає в наявності або відсутності загроз, їх активності, успішності протидії та кількості загроз, які впливають на систему. Контроль і реагування на ці стани є важливими аспектами забезпечення функціональної безпеки технологічних процесів.

Розуміння цих шести станів допомагає визначити етапи безпеки системи та необхідні дії для забезпечення її захищеності.

можливостей для продуктивної роботи над проектами різного розміру та складності.

2.2 Взаємозв'язок між станами та переходи між ними

Взаємозв'язок між станами системи безпеки(рисунок 2.7) визначається переходами, які відбуваються в результаті зміни умов і обставин. Кожен стан може мати свої можливі переходи до інших станів, залежно від розвитку ситуації та ефективності заходів безпеки. Нижче наведено загальний опис переходів між станами системи безпеки:

a) Перехід зі стану 1 "Не існує загрози":
   1) Якщо з'являється нова загроза, система може перейти в стан 2 "Загроза існує, але не активна".
b) Перехід зі стану 2 "Загроза існує, але не активна":
   1) Якщо загроза активується і стає реалізованою, система переходить в стан 3 "Загроза існує і реалізована".
   2) Якщо загроза зникає або стає неактивною, система може повернутися до стану 1 "Не існує загрози".
c) Перехід зі стану 3 "Загроза існує і реалізована":
   1) Якщо загроза успішно відбита і безпека відновлена, система може перейти в стан 4 "Загроза існує, реалізована і успішно відбита".
   2) Якщо загроза продовжує впливати на систему, або нова загроза з'являється, система може перейти в стан 5 "Існує дві загрози, перша реалізована, друга не активна".
d) Перехід зі стану 4 "Загроза існує, реалізована і успішно відбита":
   1) Якщо загроза повертається або нова загроза з'являється, система може перейти в стан 5 "Існує дві загрози, перша реалізована, друга не активна".
e) Перехід зі стану 5 "Існує дві загрози, перша реалізована, друга не активна":
   1) Якщо друга загроза активується і стає реалізованою, система переходить в стан 6 "Існує дві загрози обидві реалізовані".

2) Якщо загрози відбиваються і безпека відновлена, система може повернутися до стану 4 "Загроза існує, реалізована і успішно відбита".

f) Перехід зі стану 6 "Існує дві загрози обидві реалізовані":

1) Якщо загрози зникають або стають неактивними, система може перейти в стан 5 "Існує дві загрози, перша реалізована, друга не активна".
2) Якщо система знаходиться в стані 6 і продовжує піддаватись впливу двох загроз, додаткові переходи можуть бути визначені в залежності від специфіки системи та прийнятих заходів безпеки.

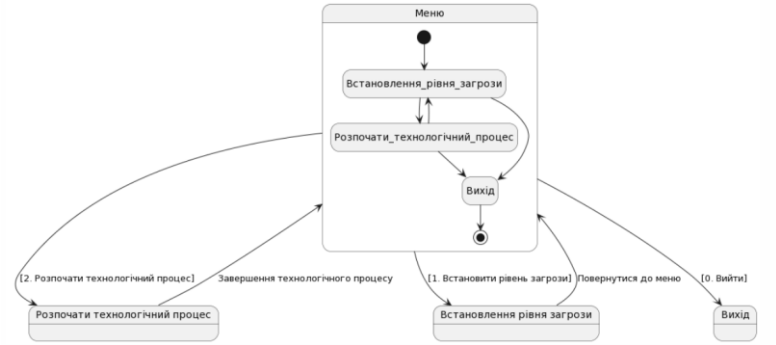

Рисунок 2.7 – Взаємозв’язок між станами системи

2.2.1 Діаграма класів

Діаграма класів відображає структуру програми, що реалізує систему функціональної безпеки. Вона включає два основних класи: "Система Функціональної Безпеки" і "Main".

a) Клас "Система Функціональної Безпеки":

1) Цей клас відповідає за функціонування системи функціональної безпеки.
2) Має внутрішнє поле "threat_level", яке вказує поточний рівень загрози.
3) Методи: "встановити Рівень Загрози(level: int)": встановлює рівень загрози для системи.

4) "показати Меню()": виводить меню на екран.
5) "обробити Вхід Меню(choice: int)": обробляє вибір користувача з меню та повертає логічне значення, яке вказує, чи продовжувати виконання програми.
6) "розпочати Технологічний Процес()": розпочинає технологічний процес залежно від рівня загрози.
7) "перевірка Безпеки()": виконує перевірку функціональної безпеки, включаючи розрахунок загрози з використанням методу Рунге-Кута та інші перевірки.
8) "розрахувати Похідну(t: float, threat: float)": обчислює похідну загрози від часу та поточного рівня загрози.
9) "знайти Лямбду()": виконує пошук параметра lambda з використанням методу золотого перетину.
10) "розрахувати Похибку(lambd: float)": розраховує похибку для заданого параметра lambda.
11) "моніторинг Та Аналіз Подій()": виконує моніторинг та аналіз безпекових подій.
12) "отримати Рівень Загрози(threat: float)": повертає рівень загрози залежно від обчисленої загрози.

b) Клас "Main":
1) Цей клас містить головний цикл програми.
2) Має внутрішнє поле "system", яке представляє екземпляр класу "Система Функціональної Безпеки".
3) Методи:

- "запустити()": ініціалізує систему та починає головний цикл програми.
- "завершити()": завершує виконання програми.

У діаграмі класів зображено залежності між цими класами та основні методи кожного класу. Вона допомагає зрозуміти структуру та функціональність програми, що реалізує систему функціональної безпеки.

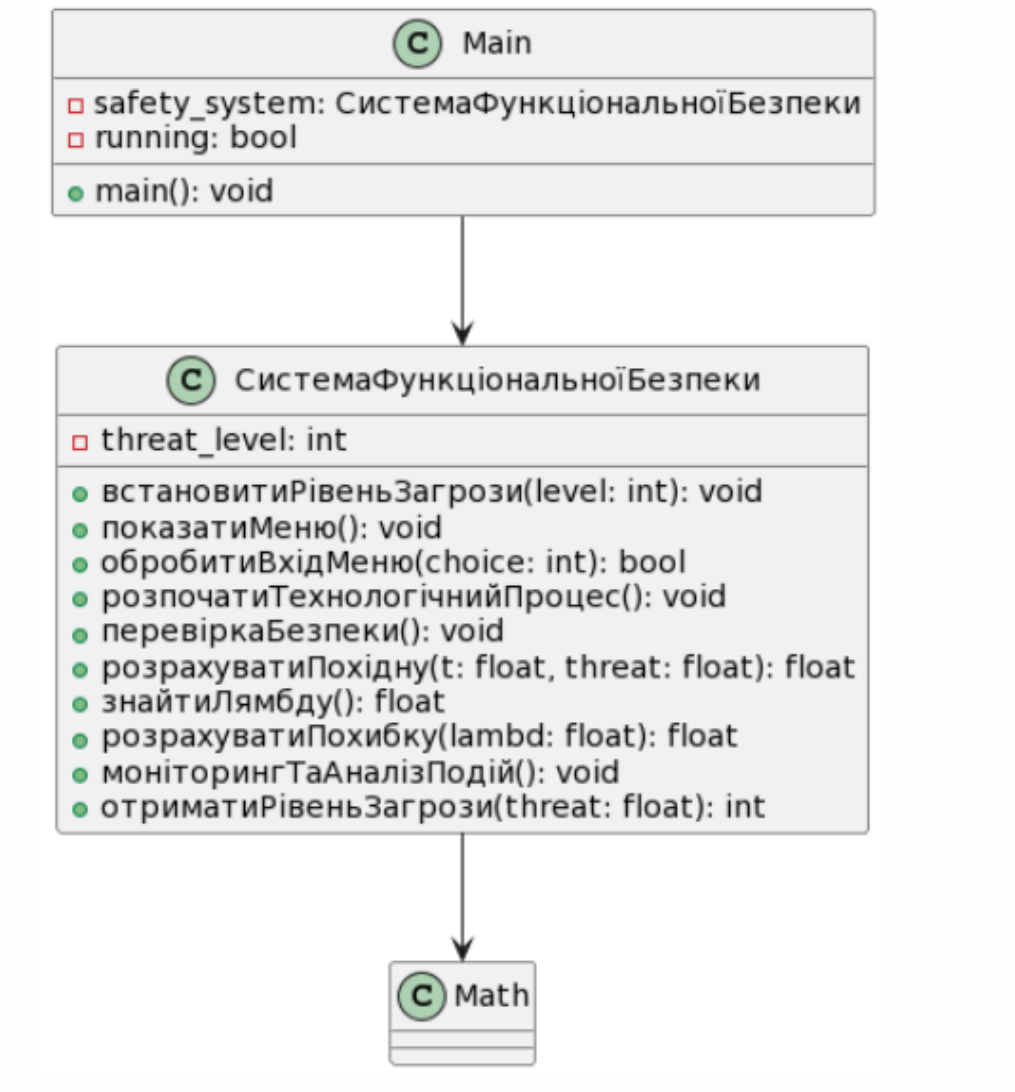

Рисунок 2.8 – Діаграма класів

2.2.2 Діаграма прецендентів

Діаграма прецедентів(рисунок 2.9) зображує основні можливості користувача і взаємодію з системою функціональної безпеки. Основним актором на діаграмі є "Користувач", який виконує дії в системі.

Система "Система Функціональної Безпеки" має три прецеденти:

1) "Встановити рівень загрози": Цей прецедент дозволяє користувачеві встановити рівень загрози для системи функціональної безпеки. Користувач може ввести числове значення рівня загрози, яке потім буде використано системою.

2) "Розпочати технологічний процес": Цей прецедент дозволяє користувачеві розпочати технологічний процес. Залежно від встановленого рівня загрози, система виводить відповідне повідомлення про стан загрози.

3) "Вийти": Цей прецедент дозволяє користувачеві завершити використання системи. Після виходу з програми, система може відображати підтверджувальне повідомлення або виконувати інші дії, пов'язані з вихідними процедурами.

Діаграма прецедентів допомагає уявити, які можливості доступні користувачу і як вони пов'язані зі структурою системи. Вона слугує основою для подальшого розроблення функціональності та деталей системи, а також для забезпечення взаєморозуміння між розробниками, дизайнерами та замовниками проекту.

Рисунок 2.9 – Діаграма прецендентів

2.2.3 Діаграма розгортання

Діаграма розгортання, що зображена нижче, описує фізичну архітектуру системи функціональної безпеки. Основні компоненти системи включають користувацький інтерфейс (UI), систему функціональної безпеки (SafetySystem) і базу даних (Database).

a) Користувацький інтерфейс (UI):
   1) Цей компонент відповідає за взаємодію з користувачем.
   2) Він може представляти собою додаток, який надає користувачеві меню для вибору команд.
   3) Користувач може вводити команди та отримувати відповіді з системи функціональної безпеки.

b) Система функціональної безпеки (SafetySystem):
   1) Цей компонент виконує логіку системи функціональної безпеки.
   2) Він має методи для встановлення рівня загрози, показу меню, обробки введення користувача, розпочатку технологічного процесу та перевірки безпеки.
   3) Він також взаємодіє з базою даних для збереження та отримання необхідної інформації.

c) База даних (Database):
   1) Цей компонент забезпечує зберігання даних, необхідних для системи функціональної безпеки(не обов’язково присутній).
   2) Він може використовувати будь-яку форму бази даних, таку як реляційну базу даних або NoSQL-сховище.

Діаграма розгортання допомагає візуалізувати фізичну архітектуру системи та показати, як компоненти взаємодіють між собою. У цьому випадку, користувацький інтерфейс взаємодіє з системою функціональної безпеки, а також система функціональної безпеки взаємодіє з базою даних для отримання необхідної інформації.

Рисунок 2.101 – Діаграма розгортання

2.2.4 Діаграма випадків використання.

Діаграма випадків використання (Use Case Diagram) (рис 2.11) відображає основні функції та взаємодію з класом FunctionalSafetySystem. Діаграма допомагає зрозуміти, як система взаємодіє з користувачем і які функціональні можливості доступні.

У цій діаграмі присутні наступні випадки використання:

1) Показати меню: Цей випадок використання відображає опцію показу меню для користувача. Користувач може переглянути доступні опції, що надаються системою.

2) Обробити вибір з меню: Цей випадок використання відображає обробку вибору користувача з меню. Після показу меню користувач може вибрати одну з опцій, і система обробить цей вибір.

3) Встановити рівень загрози: Цей випадок використання дозволяє користувачеві встановити рівень загрози. Користувач може ввести числове значення, що представляє рівень загрози, і система збереже цей рівень для подальшого використання.

4) Розпочати технологічний процес: Цей випадок використання описує початок технологічного процесу. Користувач може ініціювати початок процесу, і система виконає перевірку безпеки та запустить процес залежно від встановленого рівня загрози.

5) Моніторинг і аналіз безпекових подій: Цей випадок використання представляє моніторинг і аналіз безпекових подій системи. Система стежить за безпековими подіями, такими як вторгнення в систему, помилки датчиків або несанкціонований доступ, та аналізує їх для подальшої обробки.

Ця діаграма випадків використання допомагає краще зрозуміти основні функції системи та їх взаємозв'язок з користувачем. Вона відображає, як користувач може взаємодіяти з системою та як система реагує на користувацькі запити, забезпечуючи безпеку технологічного процесу.

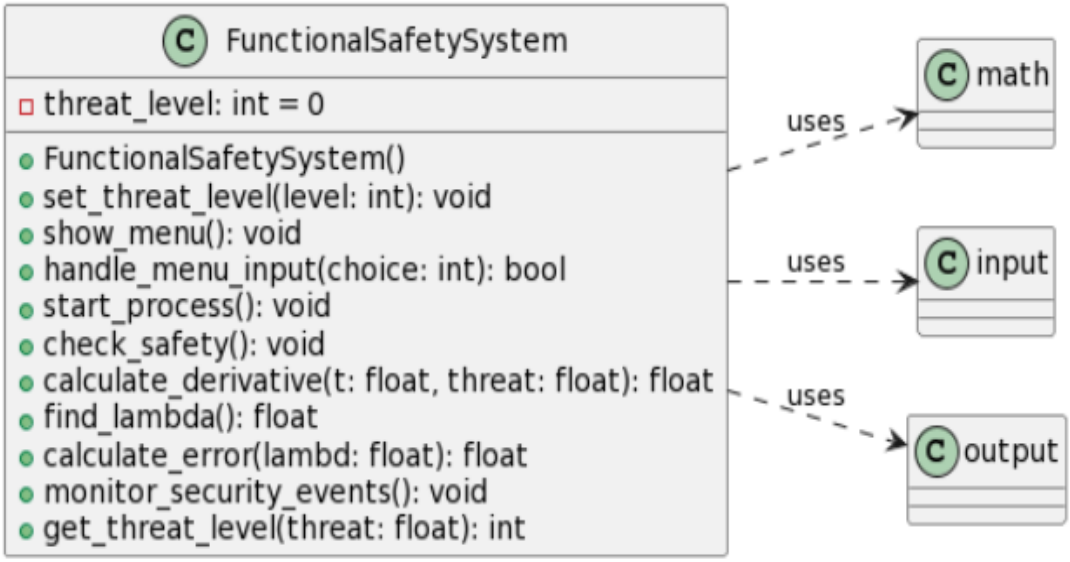

Рисунок 2.11 – Діаграма випадків використання

2.2.5 Діаграма активностей

Діаграма активностей(рисунок 2.12) відображає послідовність дій, які відбуваються у системі функціональної безпеки. Основна мета системи - виявлення та обробка загроз безпеці. Діаграма активностей описує взаємодію користувача з системою через меню, а також внутрішні дії системи, пов'язані з обробкою загроз та розпочаттям технологічного процесу.

Діаграма починається зі стартового вузла і має основний цикл, який виконується, поки система працює. На кожній ітерації циклу система показує меню з доступними опціями користувачу. Користувач може вибрати одну з опцій, і система реагує на цей вибір.

Якщо користувач обирає "Встановити рівень загрози", система вимагає ввести рівень загрози. Після введення рівня загрози система встановлює його.

Якщо користувач обирає "Розпочати технологічний процес", система виконує перевірку безпеки. У процесі перевірки безпеки система розраховує рівень загрози з використанням методу Рунге-Кута з підбором параметра lambda. Після розрахунку рівня загрози система перевіряє його значення та виводить відповідне повідомлення про розпочаття технологічного процесу.

Якщо користувач обирає "Вийти", система завершує роботу і виводить відповідне повідомлення.

Якщо користувач вводить невідому команду, система повідомляє про це і пропонує спробувати ще раз.

Діаграма активностей дозволяє зрозуміти послідовність взаємодій між користувачем та системою функціональної безпеки. Вона допомагає краще уявити роботу системи і її можливості щодо виявлення загроз та контролю над технологічним процесом.

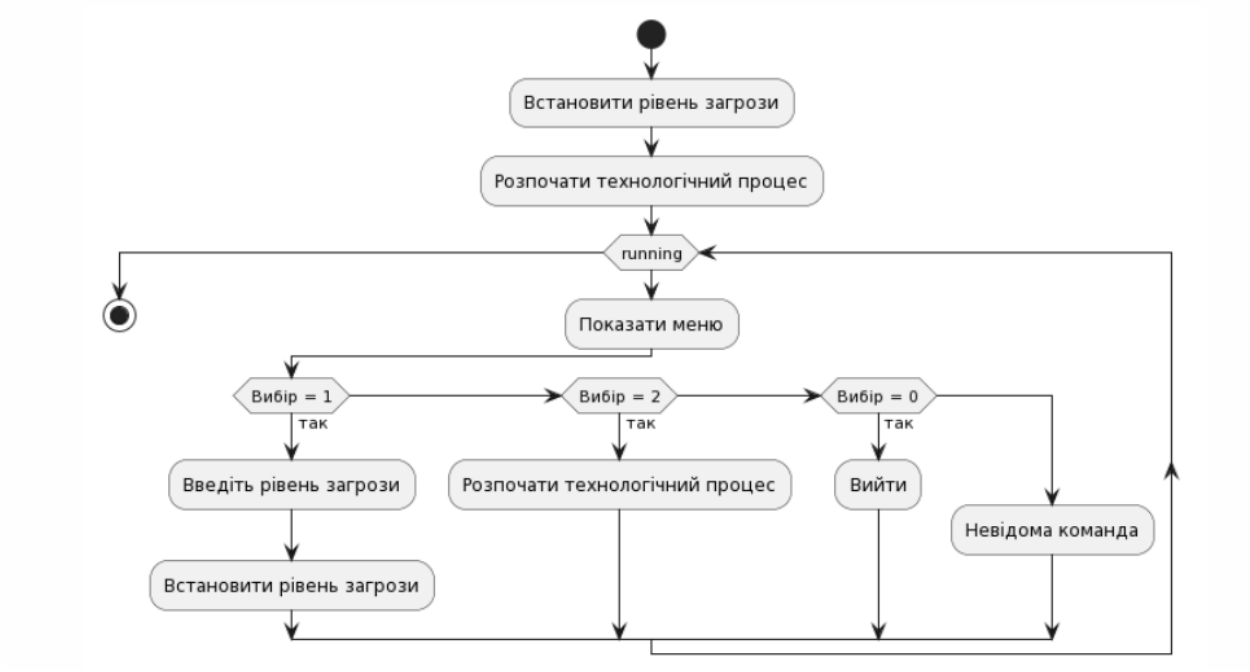

Рисунок 2.12 – Діаграма випадків використання

2.3 Алгоритми та методи управління безпекою

Алгоритми управління безпекою включають набір кроків та процедур, спрямованих на захист системи від потенційних загроз і забезпечення безпеки інформації. Основні алгоритми управління безпекою включають наступні етапи:

1) Ідентифікація загроз: Визначення потенційних загроз безпеці системи, аналіз вразливостей та оцінка ризиків.

2) Аналіз ризиків: Оцінка і призначення пріоритетів для ідентифікованих загроз на основі потенційного впливу.

3) Розробка стратегії безпеки: Визначення набору заходів та політик безпеки, які будуть використовуватися для запобігання, виявлення та реагування на загрози.

4) Впровадження заходів безпеки: Реалізація запланованих стратегій і заходів безпеки, включаючи встановлення технічних засобів, розробку політик, навчання персоналу та встановлення процедур.

5) Моніторинг та аудит безпеки: Систематичне спостереження та оцінка ефективності заходів безпеки, виявлення потенційних вразливостей та недоліків, а також вжиття заходів для їх усунення.

6) Реагування на інциденти: Розробка та впровадження планів реагування на інциденти безпеки, включаючи реагування на виявлені загрози, відновлення системи та розслідування подій.

7) Постійне вдосконалення: Оцінка та аналіз результатів заходів безпеки, постійне оновлення та вдосконалення стратегій, процедур, підходів та технологій з метою забезпечення збалансованого та оптимального рівня безпеки.

Ці алгоритми управління безпекою можуть варіюватися в залежності від конкретних вимог і характеристик системи, але загальний принцип полягає у систематичному підході до ідентифікації, захисту та моніторингу безпеки для забезпечення безперебійної та надійної роботи системи.

Управління безпекою технологічного процесу включає в себе різні методи, спрямовані на забезпечення безпеки і захисту технологічних систем. Основні методи управління безпекою технологічного процесу включають:

1) Аналіз ризиків: Цей метод полягає в ідентифікації потенційних загроз, оцінці ризиків і визначенні заходів для їх управління. Аналіз ризиків допомагає визначити вразливості та потенційні наслідки випадків порушення безпеки, що дозволяє прийняти відповідні заходи для їх запобігання або зменшення наслідків.

2) Впровадження технічних заходів безпеки: Цей метод включає в себе застосування технічних засобів для забезпечення безпеки технологічного процесу. Це можуть бути системи виявлення вторгнень, системи контролю доступу, вогнегасники, системи автоматичного відключення обладнання тощо.

3) Контроль доступу і ідентифікація: Цей метод використовується для контролю доступу до системи і ідентифікації користувачів які кормстуюється системою. Включає в себе використання паролів, біометричних технологій; таких як відбитки пальців; відбитки вух та сітківки ока; оскільки вони є унікальними у кожної людини, карт доступу та інших методів для забезпечення тільки авторизованого доступу до системи.

4) Навчання та свідомість персоналу: Цей метод включає в себе навчання персоналу про правила безпеки та процедури в разі виникнення небезпеки. Освіченість персоналу про потенційні загрози і методи їх управління є важливою складовою безпечного технологічного процесу. Освідченість персоналу принципово має вплив як на швидкість реагування на небезпеку так і на так необхідну часто можливість на виперед знешкодити потенційну небезпеку.

5) Резервне копіювання та відновлення: Цей метод передбачає створення резервних копій важливих даних та налаштувань системи для відновлення у випадку виникнення непередбачуваних ситуацій або випадків порушення безпеки. Backaup є необхідним елементом для правильного функціонування системи.

6) Моніторинг і виявлення вторгнень: Цей метод включає в себе постійний моніторинг системи для виявлення неправомірних дій або вторгнень. Використовуються спеціальні програми та системи для виявлення аномалій, атак та незвичайних змін у системі.

Ці методи можуть використовуватися окремо або в поєднанні залежно від потреб технологічного процесу та рівня безпеки, який необхідно забезпечити. Комплексне використання цих методів допомагає ефективно управляти безпекою технологічного процесу та запобігати можливим загрозам.

2.4 Математичні обрахунки

Приклад 1. Нехай матриця інтенсивності переходу має наступний вигляд:

$$\begin{vmatrix} -0{,}075 & 0{,}075 & 0 \\ 0{,}825 & -0{,}85 & 0{,}025 \\ 0{,}875 & 0 & -0{,}875 \\ & 0 & \end{vmatrix}$$

Побудуємо розв'язок задачі методом Рунге-Кутти четвертого порядку для моменту часу $t = 50$ секунд з кількістю кроків 4 на кожну одиницю часу (рисунок 2), використовуючи програмний пакет "MathCad".

В результаті проведених обчислень знайдені значення ймовірностей $p_0(t), p_1(t), p_2(t)$ для різних моментів часу, які представлені в табл. 2.1

Таблиця 2.1 Результати обчислень значень $p_0(t), p_1(t), p_2(t)$ для різних проміжків часу.

Значення часу t	Ймовірність знаходження системи в стані x_0	Ймовірність знаходження системи в стані x_1	Ймовірність знаходження системи в стані x_2
1	0,951	0,049	0
2	0,93	0,069	0,001
3	0,922	0,076	0,002
4	0,919	0,079	0,002
5	0,918	0,08	0,002
6	0,917	0,081	0,002
7	0,917	0,081	0,002
8	0,917	0,081	0,002
9	0,917	0,081	0,002
10	0,917	0,081	0,002
11	0,917	0,081	0,002

12	0,917	0,081	0,002
13	0,917	0,081	0,002
14	0,917	0,081	0,002
15	0,917	0,081	0,002
16	0,917	0,081	0,002
17	0,917	0,081	0,002
18	0,917	0,081	0,002
19	0,917	0,081	0,002
20	0,917	0,081	0,002
21	0,917	0,081	0,002
22	0,917	0,081	0,002
23	0,917	0,081	0,002
24	0,917	0,081	0,002
25	0,917	0,081	0,002
26	0,917	0,081	0,002
27	0,917	0,081	0,002
28	0,917	0,081	0,002
29	0,917	0,081	0,002
30	0,917	0,081	0,002
31	0,917	0,081	0,002
32	0,917	0,081	0,002
33	0,917	0,081	0,002
34	0,917	0,081	0,002
35	0,917	0,081	0,002
36	0,917	0,081	0,002
37	0,917	0,081	0,002
38	0,917	0,081	0,002
39	0,917	0,081	0,002
40	0,917	0,081	0,002
41	0,917	0,081	0,002
42	0,917	0,081	0,002
43	0,917	0,081	0,002
44	0,917	0,081	0,002
45	0,917	0,081	0,002
46	0,917	0,081	0,002

47	0,917	0,081	0,002
48	0,917	0,081	0,002
49	0,917	0,081	0,002
50	0,917	0,081	0,002

Відповідно, в момент часу $t = 50\, p_0(t) = 0{,}917, p_1(t) = 0{,}081, p_2(t) = 0{,}002$. У відповідність з даними, які приведені в таблиці 1, побудуємо графік зміни значення ймовірності знаходження системи в одному із станів в залежності вір часу t (рисунок 2.13).

Рисунок 2.13 – Графік зміни значення ймовірності знаходження системи в одному із станів в залежності вір часу t

Приклад 2. Нехай моменти переходу t системи зі стану в стан є дискретними ($t = n, n = 1, \dots, 8$), а матриця ймовірностей переходу за один крок має наступний вигляд:

$$\left|\left|p_{ij}\right|\right| = \begin{vmatrix} 0{,}9 & 0{,}1 & 0 \\ 0{,}9075 & 0 & 0{,}0025 \\ 1 & 0 & 0 \end{vmatrix}$$

Початковий розподіл: $p(0) = (1, 0, 0)$

Обчислимо ймовірності перебування системи в кожному зі станів x_i, $i = 0, 1, 2$, після 8 кроків, використовуючи пакет прикладних програм "MathCad". Результати обчислень представлені в таблиці 2.2.

Таблиця 2.2. Результати обчислень, отримані за допомогою програмного забезпечення "MathCad".

Крок експерименту	Ймовірність знаходження системи в стані x_0	Ймовірність знаходження системи в стані x_1	Ймовірність знаходження системи в стані x_2
1	0,9	0,1	0
2	0,88	0,1	0,02
3	0,864	0,108	0,028
4	0,856	0,111	0,033
5	0,852	0,113	0,035
6	0,849	0,114	0,037
7	0,848	0,115	0,038
8	0,847	0,115	0,038

За даними таблиці побудуємо графік зміни ймовірностей $p_0(t) = 0{,}917, p_1(t) = 0{,}081, p_2(t) = 0{,}002$. (рисунок 2.14).

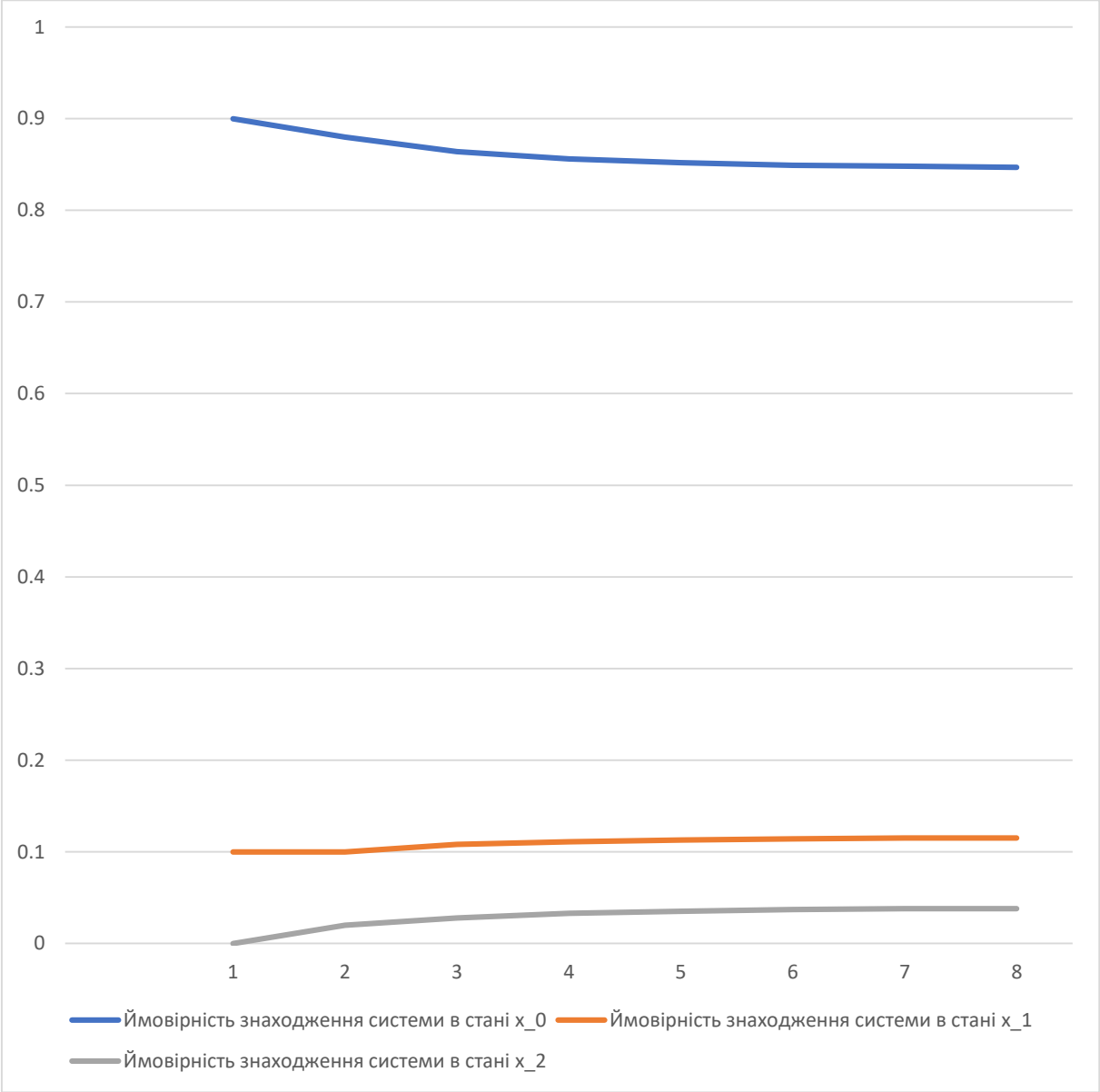

Рисунок 2.14 – Графік зміни значення ймовірності

Висновки до розділу

Висновки до розділу "Математична модель системи функціональної безпеки" можуть мати наступний зміст:

1) Математична модель системи функціональної безпеки є важливим інструментом для аналізу, проектування та управління безпековими аспектами системи. Вона дозволяє математично описати взаємодію компонентів системи та оцінити рівень безпеки.

2) Математична модель системи функціональної безпеки базується на різних методах та підходах, таких як теорія ймовірностей, теорія масового обслуговування, теорія надійності та інші. Ці методи дозволяють моделювати ймовірнісні події, оцінювати ризики та ефективність системи безпеки.

3) Розробка математичної моделі системи функціональної безпеки передбачає визначення вхідних та вихідних параметрів системи, встановлення правил та логіки взаємодії між компонентами, а також оцінку впливу зовнішніх факторів на безпеку системи.

4) Математична модель може бути використана для проведення аналізу ризиків, визначення оптимальних параметрів системи, визначення потреб у заходах з підвищення безпеки, а також для прогнозування можливих наслідків небезпечних подій.

5) Використання математичної моделі системи функціональної безпеки дозволяє зробити об'єктивні оцінки та прийняти обґрунтовані рішення з питань безпеки системи. Вона допомагає зменшити ризики, підвищити надійність та ефективність системи.

6) Врахування математичної моделі системи функціональної безпеки є важливим етапом в процесі проектування та управління безпекою системи. Вона дозволяє виявляти потенційні проблеми та вразливості системи на ранніх етапах розробки, що сприяє покращенню безпеки та зниженню витрат на подальшу експлуатацію.

Важливо підкреслити роль математичної моделі у забезпеченні безпеки системи та показати її користь у прийнятті обґрунтованих рішень.

3. РОЗРОБКА СИСТЕМИ ФУНКЦІОНАЛЬНОЇ БЕЗПЕКИ ДЛЯ ТЕХНОЛОГІЧНОГО ПРОЦЕСУ

3.1 Архітектура системи та її компоненти

Архітектура(рисунок 3.1) даного коду включає в себе клас FunctionalSafetySystem, який представляє систему функціональної безпеки. Клас містить ряд методів для взаємодії з користувачем та виконання операцій з безпекою.

1) Основні методи класу FunctionalSafetySystem:

2) __init__(self): Цей метод є конструктором класу і ініціалізує початковий рівень загрози.

3) set_threat_level(self, level): Цей метод встановлює рівень загрози відповідно до переданого значення.

4) show_menu(self): Цей метод виводить на екран меню з доступними командами.

5) handle_menu_input(self, choice): Цей метод обробляє введену користувачем команду, викликає відповідні методи залежно від вибраної команди та повертає прапорець, який вказує, чи треба продовжувати виконання програми.

6) start_process(self): Цей метод розпочинає технологічний процес, перевіряючи рівень загрози та виводячи відповідне повідомлення.

7) check_safety(self): Цей метод виконує перевірку функціональної безпеки, розраховуючи загрозу за допомогою методу Рунге-Кутта з підбором параметра lambda.

8) calculate_derivative(self, t, threat): Цей метод обчислює похідну загрози від часу та поточного рівня загрози.

9) find_lambda(self): Цей метод виконує підбір параметра lambda з використанням методу золотого перетину.

10) calculate_error(self, lambd): Цей метод обчислює похибку для заданого параметра lambda.

11) detect_threat(self): Цей метод виявляє загрозу на основі аналізу системи та стану оточення.

12) react_to_threat(self): Цей метод реагує на виявлену загрозу, виконуючи необхідні дії для забезпечення безпеки системи та процесу.

13) start(self): Цей метод починає виконання програми, виводить меню, обробляє введені користувачем команди та перевіряє наявність загрози.

У коді створюється об'єкт FunctionalSafetySystem, викликається його метод start(), який розпочинає виконання програми.

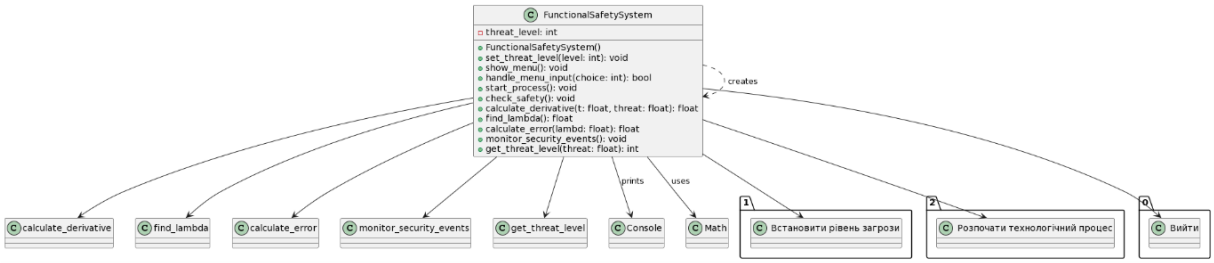

Рисунок 3.1 – Архітектура системи

3.2 Виявлення загроз та реагування на них

У даному коді виявлення загроз та реагування на них відбувається за допомогою методів detect_threat() і react_to_threat() класу FunctionalSafetySystem.

Метод detect_threat() виконує аналіз системи та стану оточення для виявлення загроз. В ньому можна реалізувати необхідні перевірки, аналіз даних з датчиків, сигналів, вимог безпеки та інших факторів. В даному коді прикладом виявлення загрози є змінна threat_detected, яка має значення False за замовчуванням.

Метод react_to_threat() виконує дії для реагування на виявлену загрозу. У даному коді прикладом реагування на загрозу є виведення повідомлення "Реагування на загрозу...". В реальному випадку метод може містити код для запуску захисних механізмів, активації системи аварійного відключення, або інших відповідних дій для забезпечення безпеки системи та процесу.

У коді метод start() викликає метод detect_threat() для перевірки наявності загрози перед виконанням дій з безпеки. Якщо загроза виявлена (якщо метод detect_threat() повертає True), то викликається метод react_to_threat(), а потім програма завершує свою роботу.

Ці методи в майбутньому можуть бути розширені та модифіковані залежно від конкретних вимог та потреб системи функціональної безпеки.

3.3 Управління ризиками та заходи безпеки

Управління ризиками та заходи безпеки в даному коді можуть бути наступними:

Визначення рівнів загроз: У коді використовується змінна threat_level, яка визначає рівень загрози. Варіанти рівнів загрози від 1 до 6 описують різні стани системи та рівні безпеки. Ви можете адаптувати ці рівні відповідно до конкретних вимог та ризиків вашої системи

Перевірка безпеки перед початком процесу: Перед початком технологічного процесу викликається метод check_safety(), який виконує перевірку функціональної безпеки системи. У даному коді ця перевірка реалізована шляхом розрахунку загрози з використанням методу Рунге-Кута та іншими обчисленнями. Ви можете розширити цей метод, додавши додаткові перевірки та аналіз для оцінки безпеки системи перед початком процесу.

Виявлення загроз: Метод detect_threat() виконує аналіз системи та стану оточення для виявлення загроз. В даному коді прикладом виявлення загрози є змінна threat_detected. Ви можете розширити цей метод, додавши додаткові перевірки, та інші механізми для виявлення потенційних загроз системі

Реагування на загрози: Метод react_to_threat() виконує дії для реагування на виявлену загрозу. У даному коді прикладом реагування на загрозу є виведення повідомлення "Реагування на загрозу...". Ви можете розширити цей метод, додавши конкретні дії для реагування на загрозу, наприклад, виконання аварійного відключення, відправлення сповіщень про безпеку, або активацію захисних механізмів.

Валідація вхідних даних: У коді можуть бути виконані перевірки вхідних даних для запобігання можливим вразливостям або неправильному виконанню системи. У даному коді виконується перевірка на те, чи передано вхідний параметр process_data та чи він має очікуваний тип даних. Ви можете додати додаткові перевірки та валідацію для інших вхідних даних.

Обробка винятків: Код містить блок try-except, який служить для обробки винятків під час виконання коду. Ви можете додати додаткові блоки except для спеціфічних винятків та виконати відповідні дії в разі виникнення помилок.

Враховуючи ці заходи безпеки та управління ризиками, можна забезпечити більшу безпеку та надійність системи. Проте, важливо пам'ятати, що конкретні заходи безпеки повинні бути вирішені залежно від конкретних потреб та характеристик системи

3.4 Моніторинг та аналіз безпекових подій

Цей код представляє клас FunctionalSafetySystem, який реалізує систему функціональної безпеки. Основний функціонал системи включає встановлення рівня загрози, запуск технологічного процесу, перевірку безпеки та моніторинг безпекових подій.

Код містить методи для взаємодії з користувачем, такі як show_menu, handle_menu_input, set_threat_level та start_process. Метод show_menu виводить на екран меню з доступними опціями. Метод handle_menu_input обробляє введення користувача та викликає відповідні методи залежно від обраної опції. Метод set_threat_level встановлює рівень загрози в системі. Метод start_process запускає технологічний процес з урахуванням встановленого рівня загрози.

Також у класі є методи для перевірки безпеки, такі як check_safety, calculate_derivative та find_lambda. Метод check_safety виконує перевірку функціональної безпеки, розраховує рівень загрози та викликає метод monitor_security_events для моніторингу безпекових подій. Метод calculate_derivative обчислює похідну загрози від часу. Метод find_lambda використовує метод золотого перетину для підбору параметра lambda.

Також у класі є методи для моніторингу та аналізу безпекових подій, такі як monitor_security_events, get_security_events та analyze_security_event. Метод monitor_security_events отримує безпекові події зі стану системи, а метод analyze_security_event аналізує конкретну безпекову подію.

Користувачу виводиться меню, де він може встановити рівень загрози або розпочати технологічний процес. Залежно від введеного вибору виконуються відповідні дії.

3.5 Робота програми

Програма має меню, де дозволяє обрати рівень загрози, або відразу розпочати технологічний процес(рисунок 3.2)

```
Меню:
1. Встановити рівень загрози
2. Розпочати технологічний процес
0. Вийти
Введіть н?мер команди:
```

Рисунок 3.2 – Меню програми

Якщо ввести неправильну команду, то програма видає відповідне повідомлення про помилку(рисунок 3.3).

```
Меню:
1. Встановити рівень загрози
2. Розпочати технологічний процес
0. Вийти
Введіть н?мер команди: 10
Невідома команда. Будь ласка, спробуйте ще раз.
```

Рисунок 3.3 – Повідомлення про неправильну команду

Якщо обрати команду «Вийти», то програма закінчить свою роботу та виведе відповідне повідомлення(рисунок 3.4)

```
Меню:
1. Встановити рівень загрози
2. Розпочати технологічний процес
0. Вийти
Введіть н?мер команди: 0
Дякую за використання системи функціональної безпеки. До зустрічі!
```

Рисунок 3.4 – Достроковий вихід з програми

При виборі команди «Встановити рівень безпеки» програма пропонує вказати рівень безпеки(рисунок 3.5)

```
Меню:
1. Встановити рівень загрози
2. Розпочати технологічний процес
0. Вийти
Введіть н◆мер команди: 1
Введіть рівень загрози (ціле число):
```

Рисунок 3.5 – Встановлення рівня загрози

Після встановлення рівня загрози, програма повертається до меню і пропонує знову обрати команду(рисунок 3.6)

```
Меню:
1. Встановити рівень загрози
2. Розпочати технологічний процес
0. Вийти
Введіть н◆мер команди: 1
Введіть рівень загрози (ціле число): 5
Меню:
1. Встановити рівень загрози
2. Розпочати технологічний процес
0. Вийти
Введіть номер команди:
```

Рисунок 3.6 – Встановлено рівень загрози

Після встановлення рівня загрози, потрібно почати технологічний процес, де система буде вказувати конкретні типи загрози(рисунок 3.7)

```
Меню:
1. Встановити рівень загрози
2. Розпочати технологічний процес
0. Вийти
Введіть н◆мер команди: 1
Введіть рівень загрози (ціле число): 5
Меню:
1. Встановити рівень загрози
2. Розпочати технологічний процес
0. Вийти
Введіть номер команди: 2
Зареєстровано по◆ію: Виявлено вторгнення в систему
Зареєстровано по◆ію: Помилка в роботі датчика
Зареєстровано подію: Н◆санкціонований доступ
Технологічний процес розпочато. Загроза існує і реалізована.
```

Рисунок 3.7 – Рішення програми

ВИСНОВКИ

Система функціональної безпеки технологічного процесу є важливим елементом в забезпеченні безпеки та надійності промислових систем. Вона дозволяє виявляти, контролювати та управляти потенційними загрозами та небезпеками, що можуть виникнути під час технологічного процесу.

Основна функція системи полягає в перевірці безпеки, аналізі потенційних ризиків та моніторингу безпекових подій. Це досягається шляхом визначення рівня загрози, що базується на аналізі параметрів системи та її оточення. Рівень загрози може варіюватися від нуля (відсутність загрози) до максимального рівня, який може призвести до зупинки технологічного процесу або навіть призвести до аварії

Система функціональної безпеки зазвичай має інтерактивний інтерфейс для взаємодії з оператором. Це дозволяє оператору встановлювати рівень загрози, запускати технологічний процес та отримувати повідомлення про стан безпеки.

Одним з ключових аспектів системи є моніторинг безпекових подій. Система відстежує події, що відбуваються в системі та отримує дані з датчиків безпеки. Ці дані аналізуються для виявлення потенційних небезпек та прийняття відповідних заходів.

У системі функціональної безпеки часто використовуються різні методи та алгоритми для оцінки рівня загрози та аналізу безпекових подій. Наприклад, метод Рунге-Кута може використовуватися для розрахунку рівня загрози з використанням певних параметрів та динамічних змінних. Метод золотого перетину може бути використаний для підбору оптимального параметра системи.

Ефективне застосування математичної моделі системи функціональної безпеки дозволяє зменшити імовірність виникнення аварійних ситуацій та негативних наслідків, що можуть поставити під загрозу безпеку персоналу та стабільну роботу технологічного процесу.

Математичні розрахунки, здійснені за допомогою програмного забезпечення, дозволяють отримати точні дані про рівень безпеки системи та зробити об'єктивні висновки щодо потенційних ризиків та необхідних заходів з підвищення безпеки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. V.S. Kharchenko, "Guarantee and guarantee systems: elements of methodology", Radio-electronic and computer systems, № 5, p. 7-19, 2006.
2. Babash AV, Shankin GP Cryptography. Edited by VP Sherstyuka, EA Primenko, - M .: SOLON-R, 2002. - 512p.
3. G.N. Gulak, “Modeling at the stage of assessing the security of confidential information encoders”, Scientific and practical journal “Modern special equipment”, № 1 (24), p. 73-81, 2011
4. G.M. Gulak, "Characteristics of dangerous failures of means that implement steganographic methods of information transformation", Scientific and Technical Journal "Information Protection", № 1 (42), p. 56-59, 2009.
5. Gulak G., Kovalchuk L., "Different approaches to the definition of random sequences", Scientific and technical collection "Legal, regulatory and metrological support of information security in Ukraine", № 3, - K., -2001, - p. .127-133.
6. Gulak GM, Mukhachev VA, Khoroshko VO, Yaremchuk YE Fundamentals of cryptographic information protection: a textbook. V .: VNTU, 2011. 198p.
7. A.G. Conheim Fundamentals of Cryptography. Per. with English - M: Radio and communication, 1987. - 412p.
8. E.L. Bauer's incomprehensible knowledge. Methods and maxims of cryptology. Two extensions. Springer. 1991. 472p.
9. Elias P., “List decoding for noisy channels”, Proceedings of WESCON Conv. Rec., 1957. ó P. 94ó 104
10. Wozenkraft J.M., “List decoding,” Quart. Progr. Rep., Res. Lab. Electron. - MIT. Cambridge, 1958. Vol. 48.

11. Guruswami V., Hastad J., Sudan M., Zuckerman D., “Combinatorial bounds for list decoding”, IEEE Trans. on Inform. Theory. - 2002. - Vol. IT-48 (5). - P. 1021 - 1034.
12. Ekdahl P., Johansson T., “Another attack on A5 / 1”, IEEE Trans. on Inform. Theory. - 2003. - Vol. IT-49 (1). - P. 284 - 289
13. Logachev OA, Salnikov AA, Yashchenko VV Boolean functions in coding theory and cryptology. - M .: МЦНМО, 2004. - 470 с.
14. Dumer II, Kabatyansky GA, Tavernier S., "List decoding of first-order Reed-Muller binary codes", Problems of information transfer. - 2007. - V. 43. № 3. - p. 66 - 74.
15. Korniyenko B., Galata L., Kozuberda O. Modeling of security and risk assessment in information and communication system. Sciences of Europe. – 2016. – V. 2. – No 2 (2). – P. 61 -63
16. Корнієнко Б.Я., Галата Л.П. Оптимізація системи захисту інформації корпоративної мережі. Математичне та комп'ютерне моделювання. Серія: Технічні науки, Випуск 19, 2019. - С. 56-62
17. Korniyenko B. The classification of information technologies and control systems. International scientific journal. – 2016. –№ 2. – P. 78 - 81.
18. Корнієнко Б.Я. Інформаційні технології оптимального управління виробництвом мінеральних добрив :монографія. – К.: Вид-во Аграр Медіа Груп, 2014. – 288 с.
19. Korniyenko B., Galata L., Ladieva L. Security Estimation of the Simulation Polygon for the Protection of Critical Information Resources / B. Korniyenko, //CEUR Workshop Proceedings, Selected Papers of the XVIII International Scientific and Practical Conference "Information Technologies and Security" (ITS 2018) Kyiv, Ukraine, November 27, 2018, Vol-2318, - P. 176-187, urn:nbn:de:0074-2318-4

20. Корнієнко Б.Я. Дослідження імітаційного полігону захисту критичних інформаційних ресурсів методом IRISK. Моделювання та інформаційні технології. 2018. Вип. 83. С. 34-41.
21. Корнієнко Б.Я. Побудова та тестування імітаційного полігону захисту критичних інформаційних ресурсів. Наукоємні технології. 2017. № 4 (36). С. 316 - 322.
22. Korniyenko B., Yudin A., Galata L. Risk estimation of information system.Wschodnioeuropejskie Czasopismo Naukowe. 2016. № 5. P. 35 - 40.
23. Корнієнко Б.Я., Юдін О.К., Снігур О.С.Безпека аутентифікації у web-ресурсах. Захист інформації. 2012. № 1 (54). С. 20 -25. DOI: 10.18372/2410-7840.14.2056 (ukr).
24. Корнієнко Б.Я., Максімов Ю.О., Марутовська Н.М. Прикладні програми управління інформаційними ризиками. Захист інформації. 2012. № 4 (57). С. 60 – 64. DOI: 10.18372/2410-7840.14.3493 (ukr).
25. Galata, L., Korniyenko, B., Yudin, A.: Research of the simulation polygon for the protection of critical information resources. In: CEUR Workshop Proceedings, Information Technologies and Security, Selected Papers of the XVII International Scientific and Practical Conference on Information Technologies and Security (ITS 2017), 30 Nov 2017, Kyiv, Ukraine. vol. 2067. pp. 23–31., urn:nbn:de:0074-2067-8.
26. Korniyenko B., Galata L. Implementation of the information resources protection based on the CentOS operating system. Conference Proceedings of 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON -2019) July 2 – 6, 2019, Lviv, Ukraine. - pp. 1007-1011.
27. Галата Л.П., Корнієнко Б.Я., Заболотний В.В. Математична модель протидії загрозам у системі захисту критичних інформаційних ресурсів. Наукоємні технології, Том 43, № 3, 2019. – С. 300 – 306.

28. Корнієнко Б.Я. Modeling of information security system in computer network. Безпека інформаційних систем і технологій, Том №1 (1), 2019. – С.36-41.
29. Korniyenko B., Galata L., Ladieva L. Research of Information Protection System of Corporate Network Based on GNS3. Conference Proceedings of 2019 IEEE Inernational Conference on Advanced Trends in Information Theory (IEEE ATIT -2019) Dezember 18 – 20, 2019, Kyiv, Ukraine. - pp. 244-248.
30. Korniyenko B., Galata L., Ladieva L. Mathematical model of threats resistance in the critical information resources protection system. CEUR Workshop Proceedings, Selected Papers of the XIX International Scientific and Practical Conference "Information Technologies and Security" (ITS 2019) Kyiv, Ukraine, November 28, 2019. Vol-2577. P.281-291.
31. Корниенко Б.Я. Кибернетическая безопасность – операционные системы и протоколы. ISBN 978-3-330-08397-4, LAMBERT Academic Publishing, Saarbrucken, Deutschland. − 2017. – 122 с.
32. Korniyenko B.Y., Galata L.P. Design and research of mathematical model for information security system in computer network. Науковий журнал «Наукоємні технології». – 2017, № 2 (34), С. 114 - 118.
33. "Functional Safety for Process Industry - Principles and Design Approaches". Функціональна безпека для промислових процесів - принципи та підходи до проектування. Авторів Йорга Френка та Вільяма М. Гілмора
34. «Functional Safety Assessment of a Technological Process in the Chemical Industry». Наукова стаття. Оцінка функціональної безпеки технологічного процесу в хімічній промисловості. Авторів Мартина Шмідта та Юргена Мюллера
35. "Functional Safety and Safety Instrumented Systems for the Process Industry Sectors" (Функціональна безпека та системи безпеки інструментів для секторів промисловості). Книга авторів Ігоря Туранського та Івана Журавля

36. IEC 61511 - Functional Safety: Safety Instrumented Systems for the Process Industry Sector. Нормативний документ. Функціональна безпека: системи безпеки інструментів для сектора промислових процесів.
37. "Analysis of Functional Safety Standards in the Oil and Gas Industry". Аналіз стандартів функціональної безпеки в нафтовій та газовій промисловості. Автора Алекса Джонсона.
38. "Integration of Functional Safety in the Design of a Chemical Process Plant"- Інтеграція функціональної безпеки в проектування хімічного заводу. Конференційна стаття. (Інтеграція функціональної безпеки в проектування хімічного заводу. Авторів Ліли Чен та Яна Ву.
39. "Cybersecurity Challenges in Functional Safety Systems for Industrial Control Systems"- Виклики кібербезпеки в системах функціональної безпеки для промислових систем управління. Журнальна стаття. Авторів Марії Родрігес та Роберта Сміта.
40. "Assessment of Functional Safety Measures for Industrial Process Plants"- Оцінка заходів функціональної безпеки для промислових процесних заводів. Дослідження. Автора Петера Мюллера.
41. "Safety Instrumented Systems - Design, Analysis, and Justification"- Системи безпеки інструментів - проектування, аналіз та обґрунтування. Книга автора Пола Галлагера.
42. "Importance of Functional Safety in the Automotive Industry". - Важливість функціональної безпеки в автомобільній промисловості. Наукова стаття автора Міхаеля Шульца.
43. "Functional Safety Management in the Oil and Gas Industry" - Управління функціональною безпекою в нафтовій та газовій промисловості. Наукова стаття авторів Емілі Браун та Девіда Сміта.
44. IEC 61508 - Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. IEC 61508 - Функціональна безпека електричних/електронних/програмованих електронних систем, пов'язаних із безпекою. Нормативний документ.

45. ANSI/ISA-84.00.01: "Functional Safety: Safety Instrumented Systems for the Process Industry Sector."

ДОДАТОК А

```
import math


class FunctionalSafetySystem:
    def __init__(self):
        self.threat_level = 0


    def set_threat_level(self, level):
        self.threat_level = level


    def show_menu(self):
        print("Меню:")
        print("1. Встановити рівень загрози")
        print("2. Розпочати технологічний процес")
        print("0. Вийти")


    def handle_menu_input(self, choice):
        if choice == 1:
            level = int(input("Введіть рівень загрози (ціле число): "))
            self.set_threat_level(level)
        elif choice == 2:
            self.start_process()
        elif choice == 0:
            print("Дякую за використання системи функціональної безпеки. До
зустрічі!")
            return False
        else:
            print("Невідома команда. Будь ласка, спробуйте ще раз.")
```

```
        return True

    def start_process(self):
        self.check_safety()

        if self.threat_level == 1:
            print("Технологічний процес розпочато. Не існує загрози.")
        elif self.threat_level == 2:
            print("Технологічний процес розпочато. Загроза існує, але не активна.")
        elif self.threat_level == 3:
            print("Технологічний процес розпочато. Загроза існує і реалізована.")
        elif self.threat_level == 4:
            print("Технологічний процес розпочато. Загроза існує, реалізована і успішно відбита.")
        elif self.threat_level == 5:
            print("Технологічний процес розпочато. Існує дві загрози, перша реалізована, друга не активна.")
        elif self.threat_level == 6:
            print("Технологічний процес розпочато. Існує дві загрози обидві реалізовані.")
        else:
            print("Увага! Небезпека! Технологічний процес не може бути розпочато.")

    def check_safety(self):
        # Виконати перевірку функціональної безпеки
        # Виконати необхідні перевірки, аналізувати стан системи,
        # перевіряти датчики, сигнали, вимоги безпеки тощо
```

```
        # Розрахунок загрози з використанням методу Рунге-Кута з підбором параметра lambda
        t = 0  # Початковий час
        h = 0.1  # Крок часу
        threat = 0  # Початкове значення загрози
        lambd = self.find_lambda()

        while t < 1:
            k1 = self.calculate_derivative(t, threat)
            k2 = self.calculate_derivative(t + h/2, threat + h/2 * k1)
            k3 = self.calculate_derivative(t + h/2, threat + h/2 * k2)
            k4 = self.calculate_derivative(t + h, threat + h * k3)

            threat += (h/6) * (k1 + 2*k2 + 2*k3 + k4)
            t += h

        self.threat_level = self.get_threat_level(threat)

        # Моніторинг і аналіз безпекових подій
        self.monitor_security_events()

    def calculate_derivative(self, t, threat):
        # Обчислення похідної загрози від часу та поточного рівня загрози
        # Виконати необхідні розрахунки та аналіз

        # Приклад: Загроза зростає лінійно від часу
        derivative = t

        return derivative
```

```
def find_lambda(self):
    # Підбір параметра lambda з використанням методу золотого перетину

    a = 0.1  # Початкове значення проміжку
    b = 1.0  # Кінцеве значення проміжку

    # Визначення золотого числа
    golden_ratio = (1 + math.sqrt(5)) / 2

    # Розрахунок проміжних точок
    x1 = b - (b - a) / golden_ratio
    x2 = a + (b - a) / golden_ratio

    while abs(b - a) > 0.0001:
        f1 = self.calculate_error(x1)
        f2 = self.calculate_error(x2)

        if f1 < f2:
            b = x2
            x2 = x1
            x1 = b - (b - a) / golden_ratio
        else:
            a = x1
            x1 = x2
            x2 = a + (b - a) / golden_ratio

    return (a + b) / 2
```

```
    def calculate_error(self, lambd):
        # Розрахунок похибки для заданого параметра lambda
        error = abs(self.calculate_derivative(1, 0) - 4)  # Приклад

        return error

    def monitor_security_events(self):
        # Виконати моніторинг та аналіз безпекових подій
        # Враховувати сигнали, датчики, системи виявлення загроз та ін.

        # Імітація безпекових подій
        events = ["Виявлено вторгнення в систему", "Помилка в роботі датчика",
"Несанкціонований доступ"]
        for event in events:
            print("Зареєстровано подію: " + event)

        # Аналіз подій та встановлення рівня загрози
        self.threat_level += len(events)

    def get_threat_level(self, threat):
        if threat == 1:
            return 1
        elif threat == 2:
            return 2
        elif threat == 3:
            return 3
        elif threat == 4:
            return 4
```

```
        elif threat == 5:
            return 5
        elif threat == 6:
            return 6
        else:
            return 0

# Створення екземпляра системи функціональної безпеки
safety_system = FunctionalSafetySystem()

# Головний цикл програми
running = True
while running:
    safety_system.show_menu()
    choice = int(input("Введіть номер команди: "))
    running = safety_system.handle_menu_input(choice)
```