Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:
«Розробка програмного засобу для виявлення атак на промислові системи керування (ICS) з використанням автоенкодерів»

Виконав: студент 4 курсу, групи 1КІТС–21б
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
_____ Підлуцький О.А.
Керівник: д. ф. (PhD), доц. кафедри МБІС
_____ Салієва О.В.
«____» _____ 2025 р.
Рецензент: к.т.н., доц., доцент каф. ОТ
_____ Крупельницький Л.В.
прізвище та ініціали)
«____» _____ 2025 р.

Допущено до захисту
Голова секції УБ, кафедра МБІС
д.т.н., проф. Яремчук Ю. Є. _____
“____” _____2025 р.

Вінниця ВНТУ – 2025.

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
д.т.н., проф. Яремчук Ю.Є.
" 20 " березня 2025 р.

З А В Д А Н Н Я
на бакалаврську кваліфікаційну роботу студенту

Підлуцькому Олександру Андрійовичу
(прізвище, ім`я, по-батькові)

1. Тема роботи: Розробка програмного засобу для виявлення атак на промислові системи керування (ICS) з використанням автоенкодерів
Керівник роботи: д. ф. (PhD), доц. кафедри МБІС Салієва О.В.
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 20 " березня 2025 року № 97
2. Термін подання студентом роботи за тиждень до захисту
3. Вихідні дані до роботи
Метою роботи є розробка програмного засобу для виявлення атак на промислові системи керування (ICS) із використанням технології автоенкодерів для аналізу аномальної поведінки мережевого трафіку та протоколів промислових мереж.
4. Зміст текстової частини:
В першому розділі бакалаврської кваліфікаційної роботи здійснюється загальний аналіз існуючих методів та програмного забезпечення виявлення кібератак на промислові системи керування (ICS), їх переваг та недоліків, а також досліджується специфіка роботи промислових протоколів та вимоги до детектування аномалій у мережевому трафіку ICS. В другому розділі розробляється підхід до створення програми виявлення атак з використанням автоенкодерів для аналізу мережевого трафіку промислових систем керування та ідентифікації аномальної активності. У третьому розділі бакалаврської кваліфікаційної роботи здійснюється програмна реалізація системи, що автоматизує запропонований підхід до виявлення кібератак на ICS з використанням методів машинного навчання. Проводиться розробленого програмного засобу на прикладі моделювання атак на промислові системи керування. У висновках підбиваються усі результати проведеної роботи.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)
У першому розділі бакалаврської дипломної роботи наявно 2 рисунка, у другому розділі – 5 рисунка, у третьому розділі – 4 рисунка.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Салієва О.В. д. ф. (PhD), доц. кафедри МБІС		
Розділ II	Салієва О.В. д. ф. (PhD), доц. кафедри МБІС		
Розділ III	Салієва О.В. д. ф. (PhD), доц. кафедри МБІС		

7.Дата видачі завдання 20 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Визначення напрямку бакалаврської роботи, формулювання теми	21.03.2025	28.03.2025	
2	Аналіз предметної області обраної теми	28.03.2025	14.04.2025	
3	Розробка алгоритму роботи	15.04.2025	22.04.2025	
4	Написання бакалаврської роботи на основі розробленої теми	23.04.2025	16.05.2025	
5	Передзахист бакалаврської кваліфікаційної роботи	16.05.2025	26.05.2025	
6	Виправлення, уточнення, корегування бакалаврської кваліфікаційної роботи	27.05.2025	04.06.2025	
7	Захист бакалаврської кваліфікаційної роботи	09.06.2025	11.06.2025	

Студент _____ Підлуцький О. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Салієва О. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається із 70 сторінку тексту формату А4, включаючи 11 рисунків, 7 таблиць, посилання на 48 літературних джерел та 4 додатка.

У даній бакалаврській кваліфікаційній роботі досліджено та реалізовано програмний засіб для виявлення аномальної активності у промислових системах керування з використанням автоенкодерів, що дозволяє своєчасно виявляти потенційні кіберзагрози без наявності прикладів атак. Проведено комплексний аналіз специфіки функціонування промислових систем керування та сучасних методів виявлення кіберзагроз. Обґрунтовано перспективність використання автоенкодерів для детекції аномалій у ICS-середовищах, оскільки ці нейронні мережі здатні навчатися виключно на нормальних даних та ефективно виявляти відхилення, що можуть свідчити про кібератаки. Розроблено та реалізовано програмний засіб з модульною архітектурою, що забезпечує гнучкість та масштабованість системи. Система включає компоненти обробки даних, навчання автоенкодера та користувацький інтерфейс з можливостями моніторингу в реальному часі. Експериментальна перевірка підтвердила ефективність розробленого підходу у виявленні різних типів аномалій у технологічних параметрах промислових систем.

Ключові слова: промислові системи керування, ICS, кібербезпека, виявлення аномалій, автоенкодери, машинне навчання, кіберзагрози, критична інфраструктура.

ABSTRACT

The bachelor's thesis consists of 70 pages of A4 text, including 11 figures, 7 tables, references to 48 literature sources and 4 appendices.

In this bachelor's thesis, a software tool for detecting anomalous activity in industrial control systems using auto-encoders was researched and implemented, which allows timely detection of potential cyber threats without the presence of examples of attacks. A comprehensive analysis of the specifics of the functioning of industrial control systems and modern methods of detecting cyber threats is carried out. The prospects of using auto-encoders for detecting anomalies in ICS environments are substantiated, since these neural networks are able to learn exclusively from normal data and effectively detect deviations that may indicate cyberattacks. A software tool with a modular architecture was developed and implemented, which ensures the flexibility and scalability of the system. The system includes data processing components, auto-encoder training, and a user interface with real-time monitoring capabilities. Experimental verification has confirmed the effectiveness of the developed approach in detecting various types of anomalies in the technological parameters of industrial systems.

Keywords: industrial control systems, ICS, cybersecurity, anomaly detection, autoencoders, machine learning, cyber threats, critical infrastructure.

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі промислові системи керування (ICS) відіграють критично важливу роль у функціонуванні енергетики, транспорту, виробництва та інших ключових галузей. Однак із зростанням цифровізації та підключенням цих систем до корпоративних мереж, вони стають вразливими до кіберзагроз. Атаки на ICS, такі як відомі інциденти з Stuxnet або атаки на енергосистему України, демонструють, що наслідки можуть бути катастрофічними – від фінансових втрат до загрози життю людей [1].

Традиційні методи кіберзахисту, орієнтовані на звичайні IT-системи, часто виявляються неефективними для ICS. Це пов’язано з особливостями промислового обладнання, специфічними протоколами зв’язку та високими вимогами до безперебійної роботи. Багато ICS працюють у режимі реального часу, і будь-які затримки в аналізі безпеки можуть призвести до серйозних проблем. Тому постає потреба в розробці спеціалізованих інструментів, здатних виявляти аномалії та атаки, не порушуючи штатну роботу системи.

Одним із перспективних підходів у цій сфері є використання автоенкодерів – нейронних мереж, які можуть навчатися на звичайних даних роботи системи та виявляти відхилення, характерні для кібератак. На відміну від сигнатурних методів, автоенкодери не потребують попередніх знань про конкретні загрози, що робить їх особливо корисними для захисту ICS, де нові типи атак з’являються постійно. Розробка програмного засобу на основі автоенкодерів може значно підвищити рівень безпеки промислових систем, забезпечуючи своєчасне виявлення та реагування на кіберзагрози.

Метою роботи є розробка програмного засобу для виявлення аномальної активності у промислових системах керування ICS із використанням автоенкодерів, що дозволяє своєчасно виявляти потенційні загрози без наявності прикладів атак.

Задачі дослідження:

– проаналізувати специфіку функціонування промислових систем керування та вимоги до їх кібербезпеки;

– проаналізувати переваги та недоліки існуючих методів виявлення аномалій у промислових системах ICS;

– розробити підхід до виявлення аномальної активності із застосуванням автоенкодерів на промислові системи керування, навчених виключно на "нормальних" даних;

– спроєктувати архітектуру та реалізувати програмний засіб, що автоматизує процес виявлення аномалій у вхідних даних ICS;

– протестувати та оцінити точність програмного засобу на прикладі відкритого датасету, що моделює реальні ICS-сценарії;

– створити інструкцію для користувача програмного засобу.

Об'єктом дослідження є процес виявлення атак на промислові системи керування ICS з використанням автоенкодерів в умовах потенційних кіберзагроз.

Предметом дослідження є методи виявлення аномалій у поведінці промислових систем на основі автоенкодерів, натренованих на легітимних (нормальних) даних.

Новизна роботи полягає в розробці спеціалізованого підходу до виявлення атак на промислові системи керування ICS з використанням модифікованих автоенкодерів, адаптованих під специфіку промислових протоколів та умов реальної експлуатації.

Практична цінність роботи полягає у створенні ефективного інструменту для виявлення аномалій у промислових системах керування шляхом застосування автоенкодерів, які аналізують виключно нормальні дані без використання прикладів атак. Такий підхід дозволяє виявляти відхилення, що можуть вказувати на нові, раніше невідомі загрози, що робить систему більш гнучкою та адаптивною у контексті сучасних кіберризиків.

1 АНАЛІЗ ТЕОРЕТИЧНИХ ОСНОВ ВИЯВЛЕННЯ АТАК НА ПРОМИСЛОВІ СИСТЕМИ КЕРУВАННЯ (ICS)c

Перший розділ присвячений комплексному дослідженню теоретичних основ кібербезпеки промислових систем керування та сучасних підходів до виявлення кіберзагроз. У розділі розглянуто фундаментальні поняття безпеки ICS, проаналізовано типологію атак на промислові системи та їх специфічні особливості порівняно з традиційними IT-системами. Особливу увагу приділено аналізу існуючих методів виявлення аномалій, включаючи сигнатурні та поведінкові підходи, з визначенням їх переваг та обмежень у контексті промислового застосування. Детально досліджено потенціал використання автоенкодерів як перспективного інструменту для виявлення аномальної активності в ICS, розглянуто їх архітектурні особливості та специфіку навчання на нормальних даних без використання прикладів атак. Розділ також містить аналіз специфіки модельованих даних промислових систем та методів їх попередньої обробки для оптимізації процесів детекції загроз.

1.1 Базові поняття кібербезпеки на ICS та класифікація атак на промислові системи керування

Промислові системи керування (Industrial Control Systems, ICS) – це комплекс апаратних і програмних засобів, призначених для автоматизації технологічних процесів на виробництві та об'єктах критичної інфраструктури. У сучасних умовах стрімкого розвитку цифрових технологій, питання безпеки ICS набувають особливого значення, оскільки ці системи стають все більш інтегрованими з корпоративними мережами та хмарними сервісами, що суттєво розширює їхню поверхню для потенційних кібератак [2].

Системи ICS значно відрізняються від традиційних інформаційних технологій (IT), оскільки безпосередньо керують фізичними процесами та обладнанням [3]. Порушення їх функціонування може призвести не лише до фінансових втрат, а й до загроз життю та здоров'ю людей, екологічних

катастроф або пошкодження промислової інфраструктури. Історично, промислові системи розроблялися з акцентом на надійність та функціональність, а не на безпеку, тому часто працюють із застарілими операційними системами без регулярних оновлень безпеки [4].

Архітектура ICS включає SCADA-системи для цілісного контролю виробничих об'єктів, розподілені системи управління (DCS) [5] для координації складних технологічних ланцюгів, інтерфейси "людина-машина" (HMI) [6] для взаємодії персоналу з обладнанням, та спеціалізовані промислові мережі для надійного обміну даними між рівнями управління [7].

Конвергенція IT та OT, тобто інтеграція інформаційних та операційних технологій, створює нові виклики для кібербезпеки промислових систем. З одного боку, ця інтеграція надає можливості для підвищення ефективності виробництва, зниження експлуатаційних витрат і впровадження нових бізнес-моделей. З іншого боку, вона розширює поверхню атаки та збільшує кількість потенційних вразливостей [8].

Кібербезпека ICS має специфічні особливості, що відрізняють її від традиційної IT-безпеки: пріоритет безперервності процесів над конфіденційністю даних, обмежені ресурси пристроїв, довгий життєвий цикл обладнання, критичність часових характеристик, відсутність можливості регулярних оновлень та використання специфічних промислових протоколів без вбудованих механізмів безпеки [9].

Серія кібератак на системи управління інфраструктури електропостачання України у 2015-2016 роках стала безпрецедентною подією – першою у світі документально підтвердженою успішною кібератакою на енергосистему [10]. 23 грудня 2015 року 225 тисяч споживачів "Прикарпаттяобленерго" залишилися без електропостачання, що стало сигналом тривоги для світової спільноти щодо вразливості критичної інфраструктури. Цей інцидент наочно продемонстрував, що традиційні підходи до захисту критичної інфраструктури потребують кардинального перегляду та модернізації. Одним з можливих рішень є саме

автоенкодери, які автоматично визначають підозрілу активність на основі аналізу показників [11].

Сучасні енергетичні мережі складаються з фізичної інфраструктури (генеруючі потужності, трансформаторні підстанції, мережі електропередач) та кібернетичної інфраструктури (SCADA-системи, DCS, програмовані логічні контролери) [12]. Важливим аспектом є відокремлення ICS від корпоративних IT-систем, що часто ігнорується на практиці, створюючи додаткові уразливості [13].

Сучасні практики захисту базуються на двох стратегіях: реактивній (виявлення атаки після її завершення) та проактивній (виявлення загрози до її реалізації) [14]. На практиці переважає реактивний підхід, де засоби захисту базуються на заздалегідь визначених критеріях, а реагування відбувається вже після завершення інцидентів [15].

Значним джерелом загроз є корпоративні IT-системи, через які можуть здійснюватися атаки за допомогою соціальної інженерії, використання особистих пристроїв співробітників або компрометації облікових даних [16]. Вектор атаки в цьому випадку може включати соціальну інженерію (фішинг), використання особистих пристроїв співробітників (Bring Your Own Device, BYOD), а також компрометацію облікових даних для доступу до SCADA-середовища [17].

Центри операцій кібербезпеки (SOC / CSOC) відіграють ключову роль у сучасних системах захисту, інтегруючи інструменти SIEM (Security Information and Event Management) для автоматизації моніторингу, аналізу та реагування. Однак у вітчизняній нормативній базі відсутні чіткі вимоги щодо організації захисту в режимі реального часу (рис. 1.1) [18].

Конвергенція IT та OT створює нові виклики для кібербезпеки промислових систем, з одного боку надаючи можливості для підвищення ефективності виробництва, з іншого – розширюючи поверхню атаки та збільшуючи кількість потенційних вразливостей, що підкреслює необхідність

розвитку інноваційних методів захисту, зокрема з використанням автоенкодерів для автоматичного виявлення підозрілої активності.

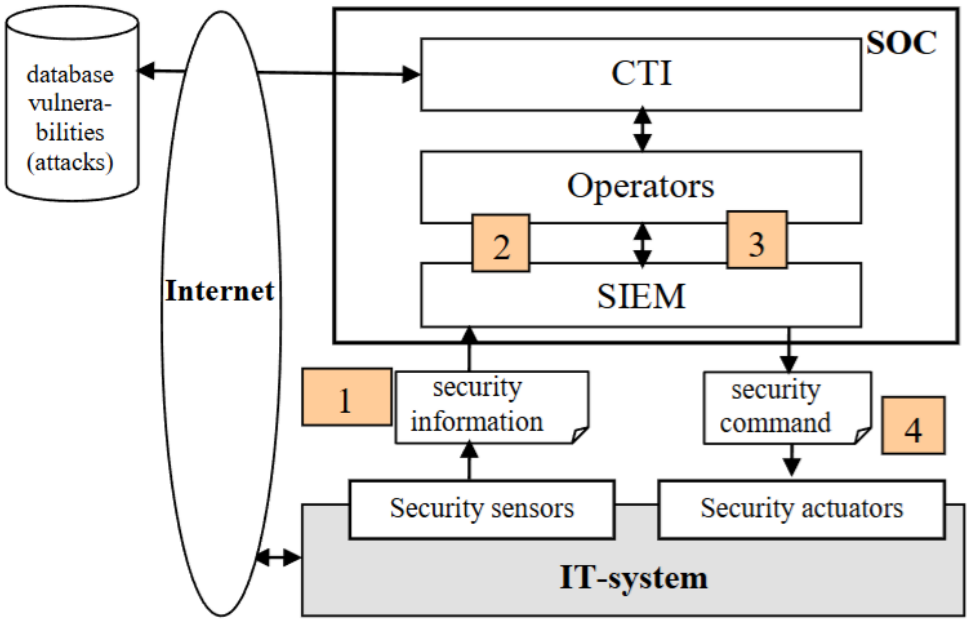


Рисунок 1.1 – Структура та основні інформаційні процеси системи оперативного кіберзахисту

Класифікація атак на ICS здійснюється за вектором проникнення, що визначає спосіб, за допомогою якого зловмисник отримує доступ до системи [19]. Основними векторами атак є незахищені інтерфейси (веб-панелі керування, відкриті API хмарних SCADA-систем, застарілі протоколи Modbus та DNP3), підключені пристрої Інтернету речей (IIoT) з вразливостями в прошивках, корпоративна IT-інфраструктура через фішингові кампанії та особисті пристрої співробітників, а також хмарне середовище з можливими DDoS-атаками та компрометацією віртуалізованих середовищ [20].

За характером впливу атаки поділяються на три основні категорії відповідно до тріади CIA (Confidentiality, Integrity, Availability) [21]. Порушення доступності проявляється у DDoS-атаках на мережеві компоненти, блокуванні роботи контролерів або виведенні з ладу SCADA-серверів. Атаки на цілісність включають маніпуляцію показниками датчиків, підміну керуючих команд у протоколах зв'язку та зловмисне оновлення прошивок промислових пристроїв.

Порушення конфіденційності передбачає несанкціоноване отримання технологічних параметрів через незашифровані канали або облікових даних через вразливості в програмному забезпеченні [22].

За рівнем складності атаки класифікуються від простих до високотехнологічних. Прості загрози реалізуються через базові віруси, скрипти або шкідливі програми, що можуть потрапити до ICS через USB-носії чи вразливі мережеві сегменти. Цільовані APT-атаки (Advanced Persistent Threats) вирізняються високим рівнем підготовки та тривалістю виконання, як у випадках Stuxnet чи Triton, що поєднують IT- та OT-інструменти для стратегічного саботажу або шпигунства [23].

Особливу категорію становлять фізико-кібернетичні атаки, які мають наслідки не лише в цифровому, а й у фізичному середовищі. Такі атаки можуть призводити до виведення з ладу обладнання через зміну параметрів його роботи, що потенційно може спричинити техногенні інциденти з загрозою для життя людей та довкілля. Ця категорія атак підкреслює критичну важливість надійного захисту промислових систем керування.

Систематизація типів атак на ICS дозволяє формувати ефективні стратегії кіберзахисту, адаптовані до специфіки промислових об'єктів. Розуміння векторів проникнення, характеру впливу, рівня складності та потенційних наслідків атак є фундаментом для побудови надійних механізмів захисту та розробки проактивних систем виявлення загроз, включаючи використання сучасних технологій машинного навчання, зокрема автоенкодерів.

1.2 Аналіз існуючих методів виявлення аномалій на промислових системах керування (signature-based, anomaly-based)

У контексті промислових систем керування завдання своєчасного виявлення аномалій є критично важливим для забезпечення безпеки технологічних процесів та запобігання аварійним ситуаціям. У сучасній практиці функціонують два основних підходи: сигнатурний (signature-based) та

аномальний (anomaly-based) [24]. Сигнатурний метод базується на пошуку збігів між вхідними даними та відомими шаблонами атак у базах сигнатур, забезпечуючи високу точність при виявленні вже відомих загроз та низький рівень хибнопозитивних спрацювань (FPR < 1%) з часом реакції 0.01-0.1 секунди, проте його ефективність значно знижується при невідомих або модифікованих атаках (zero-day) [25].

Процес виявлення починається з того, що система безпеки аналізує потік даних у реальному часі або у вигляді логів. Дані можуть містити заголовки пакетів, вміст мережевих запитів, системні виклики, команди, що виконуються, або навіть вміст файлів. Потім кожен фрагмент інформації порівнюється з базою сигнатур. Сигнатура, по суті, є детальним описом шаблону атаки, який може включати послідовність байтів, регулярні вирази, IP-адреси, порти, специфічні набори команд або певні характеристики поведінки. Формально, нехай D(t) – деяке вхідне спостереження в момент часу t, а $S = \{s_1, s_2, ..., s_n\}$ – множина сигнатур. Процедура виявлення визначається як пошук такого $s_i + S$, що $D(t) = s_i$ тобто дані відповідають хоча б одній сигнатурі [26].

З математичної точки зору, сигнатура може бути представленою у вигляді функції відповідності:

$$f_s(x) = 1, \text{ якщо x відповідає шаблону сигнатури}$$

$$f_s(x) = 0, \text{ інакше}$$

де x – вхідні дані, а $f_s(x)$ – функція відповідності сигнатурі. Після того як принаймні одна з сигнатур спрацьовує, система фіксує інцидент безпеки й викликає відповідну реакцію – логування, оповіщення, блокування запиту або навіть відключення користувача [27]. Цей метод є дуже точним при виявленні задокументованих атак, оскільки сигнатури створюються фахівцями після ретельного аналізу поведінки шкідливих програм, що дозволяє мінімізувати хибнопозитивні результати та зменшити навантаження на аналітиків кібербезпеки [28]. Аномальний підхід базується на припущенні, що нормальні дані мають певні статистичні властивості, які можна виміряти та

використовувати для виявлення відхилень. На етапі навчання метод обчислює стандартні відхилення для кожного сенсора або актуатора на основі часових рядів нормальних даних, а під час тестування класифікує вхідні дані як аномальні, якщо стандартне відхилення виходить за межі діапазону [MinS, MaxS]. Це дозволяє швидко виявляти грубі аномалії, такі як різкі стрибки значень або несподівані зміни в поведінці системи (рис. 1.2) [29].

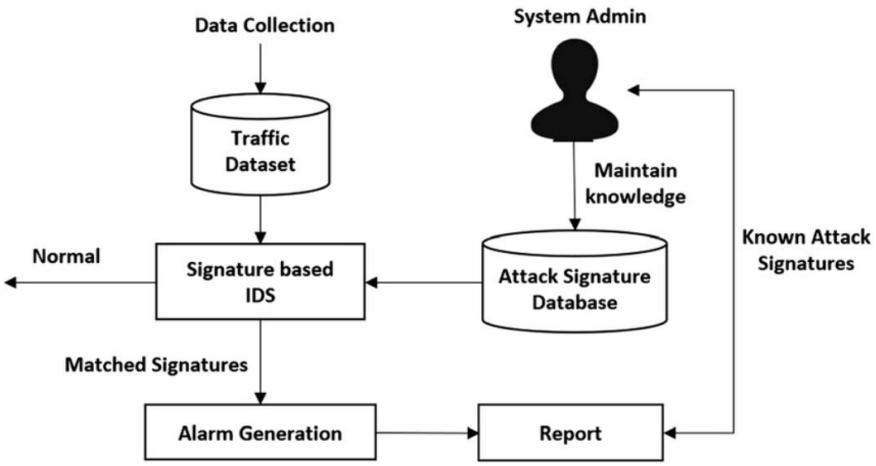


Рисунок 1.2 – Принцип роботи signature-based методу виявлення аномалій

Signature-based підхід має суттєві обмеження, насамперед нездатність виявляти нові, невідомі атаки – так звані zero-day загрози. Оскільки такі системи покладаються виключно на існуючі сигнатури, будь-яка нова загроза залишиться непоміченою доти, доки експерти не створять нову сигнатуру й не оновлять систему. Додатковим недоліком є постійна потреба в оновленні бази сигнатур та обмежена адаптивність до аномальної поведінки [30].

Натомість anomaly-based метод спирається на побудову моделей нормальної поведінки системи на основі попередньо зібраних даних. Такий підхід ефективний для виявлення zero-day атак і складних шаблонів поведінки зловмисника, однак має істотно вищий рівень хибнопозитивних спрацювань. Типовий час реакції таких систем складає від 0.1 до 1 секунди, а рівень точності при виявленні нових атак варіюється в межах 70–90% [31]. Ефективність системи значною мірою залежить від якості побудови опису допустимої поведінки та навчання системи [32].

Критичною проблемою anomaly-based систем є те, що шкідлива активність, яка відбувається в межах дозволеної моделі поведінки, може пройти непоміченою. Наприклад, складні атаки типу directory traversal, які формально відповідають протоколам, не викликають порушень у структурі трафіку. Це демонструє фундаментальну вразливість таких систем – потенційну нездатність розпізнати тонкі, протокольно-коректні атаки [33].

Робота anomaly-based методу ділиться на дві фази: навчання (створення моделі нормальної поведінки) і виявлення (виявлення відхилень). У фазі навчання використовують статистичні характеристики або методи машинного навчання, наприклад кластеризацію чи автоенкодери. Якщо розглядати, приміром, кількість запитів до певного сервера на годину, то система може обчислювати математичне сподівання μ та стандартне відхилення σ з історичних даних. Для нових спостережень перевіряється, чи потрапляє значення у допустимий інтервал:

$$x \in [\mu - k\sigma, \mu + k\sigma]$$

де x – спостережуване значення, а k – коефіцієнт чутливості (зазвичай 2 або 3) [34]. Якщо значення виходить за ці межі, воно вважається аномальним. Такий метод називається статистичним, і він добре працює в системах зі стабільними параметрами. У випадках, коли поведінка є більш складною та багатовимірною, використовують методи машинного навчання. Наприклад, система може застосовувати кластеризацію, щоб групувати нормальні шаблони поведінки, і потім обчислювати відстань нових точок до центрів кластерів. Якщо точка знаходиться надто далеко від найближчого центру, вона вважається аномалією. Використовується евклідова або косинусна відстань, наприклад:

$$D(x, \mu_i) = \sqrt{\sum_{j=1}^{n} (x_j - \mu_{ij})^2}$$

де x_j – параметри спостереження, μ_{ij} – координати центру i-го кластера [35].

Ще один популярний варіант – використання автоенкодерів з глибокого навчання. Вони навчаються стискати вхідні дані до меншого представлення, а потім відновлювати їх. Якщо вхідна поведінка є типовою, модель легко відновлює її. Якщо ж поведінка незвична, похибка реконструкції висока. Таким чином, система фіксує різницю між вхідними та відновленими даними як показник аномальності:

$$Loss = \|x - \hat{x}\|^2$$

де x – вхідні дані, $\hat{x}$ – реконструкція, а *Loss* – міра аномальності.

Після побудови моделі система переходить до фази виявлення. Вона в режимі реального часу або періодично перевіряє поточну активність у мережі чи системі, порівнюючи її з моделлю. Усе, що суттєво відрізняється від очікуваної поведінки, позначається як підозріле. Якщо така поведінка повторюється або порушує кілька характеристик одночасно, система може автоматично створювати тривогу або навіть блокувати активність.

Порівняння обох методів показує їхні принципово різні філософії: signature-based є реактивним і виявляє лише відоме, тоді як anomaly-based може виявляти нові загрози через статистичні відхилення від нормального стану системи. Signature-based система не виявить шкідливу активність з унікальним або невідомим шаблоном, що робить комбінацію обох підходів оптимальним рішенням для багаторівневої системи захисту. Тобто, якщо шкідлива активність має унікальний або невідомий шаблон, вона не буде виявлена (рис 1.3) [36].

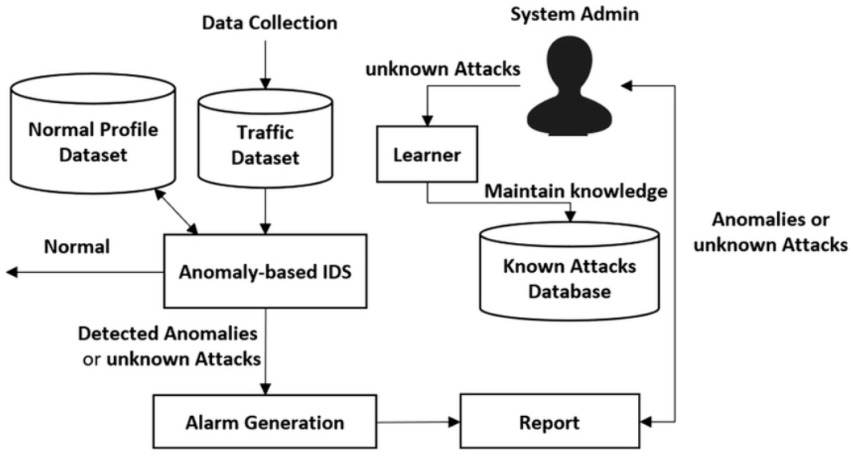


Рисунок 1.3 – Принцип роботи anomaly-based методу виявлення аномалій

Натомість anomaly-based метод є проактивним, оскільки не вимагає знання конкретних атак, а реагує на будь-яку незвичну поведінку. Це дозволяє виявляти так звані zero-day атаки або інші нетипові форми експлуатацій, які ще не описані в сигнатурній базі. З огляду на вище, наведено порівняння обох підходів за ключовими критеріями, що дозволяє об'єктивно оцінити їхню релевантність до задач виявлення аномалій у промислових мережах: (табл.1.1) [36].

Таблиця 1.1 – Порівняння методів виявлення аномалій за критеріями

Критерій	Signature-based	Anomaly-based
Виявлення відомих атак	Висока точність	Помірна ефективність
Виявлення невідомих атак	Відсутнє	Висока ймовірність
FPR (False Positive Rate)	<1%	5–30%
Обчислювальні витрати	Низькі	Високі
Реакція на Zero-day	Неможлива	Можлива
Тип середовища	Статичне	Динамічне
Необхідність оновлення	Часте оновлення баз	Періодичне перенавчання

У динамічному середовищі промислових систем обидва підходи мають обмеження: перший потребує постійного оновлення баз сігнатур, другий - періодичного перенавчання та генерує значну кількість помилкових тривог, що робить актуальними методи машинного навчання, зокрема автоенкодери, які здатні адаптуватися до нових форм загроз без залежності від наявності сігнатур.

1.3 Аналіз досвіду застосування автоенкодерів та інших нейромереж для виявлення аномалій на промислових системах керування

У сучасних підходах до виявлення аномалій у промислових системах керування (ICS) активно застосовуються різноманітні архітектури штучних нейронних мереж: багатошарові перцептрони (MLP), згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN), автоенкодери (AE), графові нейронні мережі (GNN) та їхні комбінації. Згорткові нейронні мережі (ResNet,

VGG) демонструють високу ефективність у задачах класифікації зображень, проте в контексті ICS, де ключову роль відіграють послідовні числові дані, їхня ефективність обмежена: точність виявлення аномалій коливається в межах 70–78%, при цьому потреба в GPU-пам'яті перевищує 4 ГБ, а час навчання складає понад 10 годин [37].

Рекурентні нейронні мережі (LSTM, GRU) краще пристосовані до обробки часових рядів і забезпечують вищу точність (до 84%) при аналізі динамічних змін, однак мають суттєві обмеження: нестабільність навчання, залежність від гіперпараметрів, значні витрати часу (понад 12 годин) та проблему затухаючого градієнта. Гібридні архітектури (CNN-LSTM, LSTM-AE) демонструють потенційно кращу точність (85–88%), але з значним ускладненням обчислювальної моделі: необхідна пам'ять понад 6 ГБ, час навчання перевищує 15 годин, що робить їх менш придатними для впровадження в реальному часі через складність налаштування та важкість інтерпретації результатів [38].

Графові нейронні мережі (GNN) забезпечують перспективу в задачах із чітко визначеними топологіями, однак для більшості промислових систем така графова структура недоступна або динамічно змінюється. Точність виявлення аномалій у GNN досягає лише 75%, а потреби в ресурсах у 1.5–2 рази вищі, ніж у MLP-AE. Висока складність реалізації та залежність від повноти графової інформації обмежують практичне застосування GNN в ICS, що підтверджує необхідність порівняльного аналізу архітектур з точки зору сумісності з ICS, вимог до обчислювальних ресурсів, гнучкості та складності впровадження (табл. 1.2) [39].

Таблиця 1.2 –Порівняння нейронних мереж для задач виявлення аномалій у ICS

Архітектура	Сумісність з ICS	Час навчання (год)	Пам’ять (ГБ RAM / GPU)	Складність впровадження	Середній час інференсу (мс)
MLP-AE	Висока	3–5	2–4 / GPU	Низька	5–10
CNN	Низька	10–12	6 / GPU	Висока	16–20
RNN/LSTM	Середня	12–15	6–8 / GPU	Середня	18–30
CNN-LSTM	Середня	15–18	8 / GPU	Висока	22–29
GNN	Дуже низька	14–20	6–8 / GPU	Дуже висока	30–56

Як випливає з аналізу, автоенкодери на основі багатошарового перцептрона (MLP-AE) мають найвищу сумісність із промисловими системами керування. Вони демонструють точність виявлення аномалій до 90 % при порівняно низьких вимогах до ресурсів: достатньо CPU з 4 ядрами та <2 ГБ оперативної пам’яті. Навчання моделі MLP-AE триває менш як 5 годин на стандартних наборах даних, а час інференсу в середньому не перевищує 10 мс на один запис, що дозволяє впроваджувати такі рішення в режимі реального часу.

Нижче наведено короткий аналітичний огляд ключових проблем кожної альтернативної архітектури, що перешкоджають їх повноцінному впровадженню у ICS-середовищах (табл. 1.3) [40].

Таблиця 1.3 – Ключові недоліки альтернативних архітектур нейромереж у контексті ICS

Архітектура	Основні недоліки	
CNN	Не адаптовані до часових рядів, громіздкість	Точність <78 %, GPU ≥ 4 ГБ, інференс >15 мс
LSTM/GRU	Чутливі до гіперпараметрів, довге навчання	Час навчання >12 год, інференс >20 мс
CNN-LSTM	Високі апаратні вимоги, складність масштабування	GPU ≥ 8 ГБ, час інференсу >25 мс
GNN	Вимагають графової структури, складні в реалізації	Точність <75 %, пам’ять >6 ГБ, час навчання >14 год

Моделі на основі LSTM та GRU показують дещо вищу точність (до 84 %) на послідовнісних даних, однак страждають від надмірної чутливості до гіперпараметрів, високого часу навчання (понад 12 годин) і довгого часу інференсу (>20 мс). Гібридні архітектури на кшталт CNN-LSTM, попри потенційно вищу точність (до 88 %), потребують значних апаратних ресурсів (≥8 ГБ GPU), мають складну структуру та демонструють найвищий середній час інференсу (>25 мс). Графові нейронні мережі надають нові можливості обробки зв’язних даних, проте є вкрай складними для впровадження в реальних ICS, через необхідність побудови графових структур, недостатню точність (<75 %) та

високі вимоги до обчислювальних потужностей (пам’ять >6 ГБ, час навчання >14 год).

На цьому тлі автоенкодери на базі багатошарового перцептрона демонструють найкраще співвідношення між точністю (88–90%), швидкодією (інференс ≤10 мс), стабільністю та ресурсною ефективністю (робота на CPU з <2 ГБ RAM), що робить MLP-AE найбільш доцільним вибором з огляду на баланс точності, обчислювальної ефективності, надійності та практичної реалізованості в умовах промислових обмежень ICS

1.4 Аналіз існуючих рішень та програмних засобів для виявлення атак на промислових системах керування

Промислові системи керування (ICS) становлять основу функціонування критичної інфраструктури, що зумовлює необхідність застосування ефективних засобів виявлення та протидії кібератакам. Адаптація традиційних систем виявлення вторгнень (IDS) до вимог ICS передбачає урахування обмеженої обчислювальної потужності пристроїв, обмежень у пропускній здатності та вимог до реального часу, використовуючи основні засоби сигнатурного аналізу та виявлення аномалій.

Snort, як відкрита система IDS з базою понад 13 000 правил, має обмежену ефективність у промислових середовищах через відсутність глибокого аналізу специфічних протоколів (Modbus, DNP3) та високий рівень хибних спрацьовувань (понад 20%) у мережах з нестабільним трафіком [41]. Zeek забезпечує більш широкий функціонал завдяки скриптовій мові та здатність виявляти до 60% аномальних подій у середовищі Modbus, проте обмеження в інтеграції з фізичним шаром унеможливлюють виявлення атак на технологічні параметри.

Suricata надає багатопоточну обробку та підтримку модулів для промислових протоколів, проте її можливості також обмежуються мережевим рівнем: під час випробувань у SCADA-середовищі з Modbus-стеком система виявила лише 30% подій, пов'язаних із несанкціонованими командами, та не

змогла коректно ідентифікувати складні атаки типу «replay», що підтверджує необхідність порівняльного аналізу цих рішень за ключовими параметрами ефективності. Нижче наведено порівняльний аналіз розглянутих рішень за вказаними параметрами (табл.1.4) [42].

Таблиця 1.4 – Порівняльний аналіз адаптованих систем виявлення атак у ICS

Програмне рішення	Метод виявлення атак	Підтримка промислових протоколів	Розуміння контексту ICS	Виявлення нових типів атак	Частота хибних спрацьовувань
Snort	Сигнатурний, обмежений аналіз аномалій	До 3 протоколів через додаткові правила	Відсутнє	До 20% виявлення нових атак	до 25%
Zeek	Сигнатурний, поведінковий аналіз	До 7 протоколів через кастомні скрипти	Обмежене	Середнє (до 60%)	10–15%
Suricata	Сигнатурний, обмежений аналіз аномалій	До 5 протоколів через додаткові модулі	Відсутнє	Низьке (до 30%)	20–22%

Узагальнюючи представлені дані, адаптовані засоби IDS (Snort, Zeek, Suricata) демонструють обмежену ефективність у контексті ICS, оскільки зосереджені переважно на аналізі мережевого трафіку без доступу до інформації з фізичного рівня керування. Це призводить до неможливості виявлення складних атак на рівні логіки процесу або технологічних параметрів, а підвищена частота хибних спрацьовувань знижує їхню практичну цінність у реальному промисловому середовищі.

Критичні об'єкти інфраструктури потребують застосування високоспеціалізованих засобів виявлення атак, здатних враховувати галузеву специфіку та особливості взаємодії між компонентами. Nozomi Networks SCADAguardian є провідним засобом захисту ICS з автоматичним виявленням пристроїв та глибоким аналізом понад 100 спеціалізованих промислових протоколів (Modbus, DNP3, IEC 60870-5-104), використовуючи комбінований метод виявлення атак (сигнатурний, аномальний та поведінковий підходи) та

інтеграцію з системами SIEM, проте має високу вартість впровадження та залежність від оновлень вендора [43].

Claroty Platform підтримує понад 550 ICS-протоколів та реалізує поведінковий і аномальний аналіз на основі тактик MITRE ATT&CK for ICS, демонструючи особливу ефективність на об'єктах із розподіленою інфраструктурою, однак вимагає значних обчислювальних ресурсів і схильна до помилкових спрацювань у складних мережевих топологіях [44]. Dragos Platform вирізняється глибоким орієнтуванням на аналіз реальних інцидентів у сфері ICS, включаючи моніторинг активів та виявлення загроз на основі внутрішньої бази знань, використовуючи аномальний та поведінковий аналіз специфічних TTP, проте має складність розгортання та потребу в залученні спеціалістів.

Radiflow iSID завершує огляд спеціалізованих рішень, демонструючи фокус на промислових мережах з підтримкою ключових протоколів та можливостями глибокого пакетного аналізу, що дозволяє виявляти як відомі, так і невідомі загрози через комбінацію сигнатурного та аномального підходів, забезпечуючи баланс між функціональністю та простотою впровадження для середніх та великих промислових підприємств.Порівняльний аналіз спеціалізованих рішень для захисту ICS наведено (табл.1.5) [45].

Таблиця 1.5 – Порівняльний аналіз спеціалізованих рішень для захисту ICS

Параметр	Nozomi SCADAguardian	Claroty Platform	Dragos Platform	Radiflow iSID
Кількість підтримуваних протоколів	Більше 100	Більше 550	До 100	До 200
Методи виявлення атак	Сигнатурний, аномальний, поведінковий	Аномальний, поведінковий	TTP-аналітика, поведінковий	Сигнатурний, аномальний
Адаптація до нових атак	Висока	Висока	Висока	Середня
Інтеграція з SIEM	Так	Так	Так	Частково
Підтримка стандартів безпеки	Часткова	Повна	Повна	Повна
Частота хибних спрацьовувань	Низька	Середня	Низька	Висока

Простота розгортання	Середня	Складна	Складна	Висока

Radiflow iSID відзначається здатністю до автоматичного картографування ICS-мереж та виявлення вразливостей, відповідністю галузевим стандартам (NIST, NERC CIP, ISA/IEC 62443), підтримкою до 200 ICS-протоколів і реалізацією сигнатурного та частково аномального методів виявлення загроз, однак поступається конкурентам за глибиною поведінкового аналізу та має обмежені можливості інтеграції з зовнішніми системами SIEM та EDR [46]. Проведений аналіз засвідчує, що спеціалізовані рішення демонструють вищу ефективність у виявленні загроз ICS завдяки підтримці промислових протоколів та поведінковому аналізу, проте мають недоліки у вигляді високої вартості, складності налаштування, залежності від закритого ПЗ та обмеженої гнучкості у масштабуванні.

З урахуванням переваг і недоліків існуючих рішень, існує актуальна потреба у створенні власного програмного засобу, орієнтованого на специфіку ICS, з відкритою архітектурою та адаптованого до українських промислових умов. Такий засіб має включати підтримку критично важливих протоколів, можливість самонавчання моделей та інтерпретацію фізичних аномалій у контексті керованих процесів, забезпечуючи ефективне виявлення як відомих, так і нових загроз без залежності від баз знань виробника.

1.5 Висновки та постановка задач

Отже, у першому розділі роботи було проаналізовано теоретичні основи виявлення атак на промислові системи керування (ICS), включаючи базові поняття кібербезпеки ICS, класифікацію атак та сучасні методи виявлення аномалій. Досліджено особливості методів на основі сигнатур (signature-based) та методів на основі аномалій (anomaly-based), визначено їх переваги та обмеження. Проаналізовано потенціал застосування автоенкодерів у задачах виявлення аномалій в ICS, їх здатність навчатися реконструювати нормальну поведінку системи та виявляти відхилення. Вивчено специфіку модельованих

даних ICS як джерела для виявлення аномалій та методи їх попередньої обробки для підвищення ефективності роботи систем виявлення атак.

У процесі проведення аналізу теоретичного матеріалу, було поставлено такі подальші завдання:

– розробити підхід до виявлення аномальної активності із застосуванням автоенкодерів на промислові системи керування;

– спроєктувати архітектуру та реалізовати програмний засіб, що автоматизує процес виявлення аномалій у ICS;

– протестувати та оцінити точність програмного засобу на прикладі відкритого датасету, що моделює реальні ICS-сценарії;

– створити інструкцію для користувача програмного засобу.

Виконання усіх поставлених завдань дозволить досягнути головної мети бакалаврської кваліфікаційної роботи – розробки ефективного програмного засобу для виявлення атак на промислові системи керування з використанням автоенкодерів.

2 ПРОЄКТУВАННЯ ПРОГРАНМОГО ЗАСОБУ ДЛЯ ВИЯВЛЕННЯ АТАК НА ПРОМИСЛОВИХ СИСТЕМАХ КЕРУВАННЯ (ICS) З ВИКОРИСТАННЯМ АВТОЕНКОДЕРІВ

Другий розділ присвячений розробці концептуальних основ та архітектурного проєктування програмного засобу для виявлення кіберзагроз у промислових системах керування. У розділі обґрунтовано вибір автоенкодерів як оптимального підходу для детекції аномалій в ICS-середовищах, розглянуто їх переваги над традиційними сигнатурними методами, зокрема здатність виявляти невідомі "zero-day" атаки. Детально описано архітектуру автоенкодера з компонентами кодера та декодера, принципи його функціонування на основі аналізу помилки реконструкції даних. Запропоновано комплексний алгоритм виявлення аномалій, що включає етапи отримання, попередньої обробки та аналізу даних з ICS. Розроблено модульну архітектуру програмного засобу з чітким розділенням функціональності, схеми взаємодії компонентів системи та інтерфейси користувача, що забезпечують ефективний моніторинг та управління процесом виявлення загроз у реальному часі.

2.1 Обґрунтування розроблення підходу виявлення аномалій на промислових системах керування з використанням автоенкодеру

Сучасний стан кібербезпеки промислових систем керування характеризується зростанням кількості та складності кібератак, спрямованих на критично важливі об'єкти інфраструктури. За даними аналітичних звітів, кількість кіберінцидентів у промисловому секторі зросла на 87% порівняно з попереднім роком, при цьому 68% атак залишаються невиявленими традиційними засобами захисту протягом місяці [48]. Особливо загрозливою є тенденція появи складних багатоетапних атак типу Advanced Persistent Threat (APT), які використовують комбінацію соціальної інженерії, експлойтів нульового дня та методів латерального переміщення по мережі.

Критичність ситуації посилюється специфічними особливостями промислових систем керування, які відрізняють їх від корпоративних IT-систем. По-перше, промислові системи часто експлуатуються протягом десятиліть без можливості регулярного оновлення програмного забезпечення через необхідність забезпечення безперервності виробничих процесів. По-друге, архітектура ICS-систем історично проектувалася з пріоритетом функціональності та надійності, а не безпеки, що призвело до використання застарілих протоколів зв'язку без вбудованих механізмів шифрування та автентифікації. По-третє, інциденти кібербезпеки в промислових системах можуть призвести не лише до фінансових втрат, але й до фізичних пошкоджень обладнання, екологічних катастроф та загрози життю людей.

Аналіз існуючих підходів до захисту промислових систем керування виявляє їх суттєві обмеження в контексті сучасних загроз. Традиційні системи виявлення вторгнень на основі сигнатур демонструють ефективність лише проти відомих типів атак, що становить критичну проблему, оскільки близько 43% успішних атак використовують раніше невідомі вразливості. Системи виявлення аномалій на основі правил також мають суттєві недоліки, включаючи необхідність експертного налаштування, високий рівень хибних спрацьовувань та неспроможність адаптуваться до зміни нормальної поведінки системи з часом, а складність сучасних промислових систем робить практично неможливим створення вичерпного набору правил для всіх можливих сценаріїв роботи [49].

У цьому контексті розробка підходу виявлення аномалій на основі автоенкодера набуває особливого значення як перспективне рішення для захисту промислових систем керування. Автоенкодер представляє собою особливий тип нейронної мережі, що складається з двох основних частин: кодера (encoder), який стискає вхідні дані до меншої розмірності, та декодера (decoder), який намагається відновити початкові дані з їх стисненого представлення. Ключова особливість автоенкодера полягає в тому, що він

навчається на даних, що відображають нормальний режим роботи системи, без необхідності мати приклади аномалій або атак.

Перевага розробленого підходу полягає у здатності виявляти раніше невідомі (zero-day) атаки на основі аналізу технологічних параметрів промислових процесів – температури, тиску та рівня води. Принцип роботи базується на припущенні, що автоенкодер, навчений виключно на нормальних даних цих параметрів, буде ефективно реконструювати вхідні значення, що відповідають штатному режиму роботи обладнання, але демонструватиме високу середньоквадратичну похибку (MSE) для аномальних комбінацій параметрів, які виникають внаслідок кібератак або технічних несправностей.

Система забезпечує оперативне виявлення аномалій у режимі реального часу з оновленням показників щосекунди, що критично важливо для промислових систем, де затримка в реагуванні може призвести до каскадних відмов обладнання. Автоматизоване ведення журналу подій з детальною фіксацією всіх змін параметрів дозволяє здійснювати ретроспективний аналіз інцидентів без необхідності постійного ручного моніторингу. Економічний ефект проявляється в автоматизації процесу моніторингу, що усуває необхідність цілодобового контролю операторами та забезпечує миттєве реагування на критичні відхилення.

Розроблений підхід має суттєві переваги порівняно з традиційними методами виявлення атак на промислові системи керування. На відміну від систем на основі сигнатур, автоенкодер фокусується на виявленні відхилень від нормальної поведінки технологічних параметрів незалежно від способу реалізації атаки, використовуючи виключно дані нормального функціонування системи. Модель може бути періодично перенавчена на нових нормальних даних, що дозволяє адаптуватися до еволюції системи з часом, при цьому помилка реконструкції надає кількісну міру аномальності даних для встановлення пріоритетів реагування на виявлені загрози.

Важливо зазначити, що для ефективного функціонування запропонованого підходу ключове значення має якість навчальних даних, які повинні повно та репрезентативно відображати нормальний режим роботи системи. З цією метою в розробленому програмному засобі реалізовано модуль генерації та підготовки навчальних даних, що дозволяє створювати контрольовані набори даних для навчання автоенкодера. У контексті промислових систем керування особливо важливою є здатність системи виявлення аномалій працювати в режимі реального часу, обробляючи потоки телеметричних даних від сенсорів та контролерів. Розроблений підхід забезпечує можливість такої обробки завдяки ефективній архітектурі автоенкодера та оптимізованим алгоритмам попередньої обробки даних. Розроблений підхід передбачає навчання автоенкодера на даних, що відображають нормальний режим роботи промислової системи керування, та подальше використання навченої моделі для виявлення аномалій шляхом аналізу помилки реконструкції. Алгоритм включає етапи отримання даних з ICS, їх попередньої обробки та нормалізації, передачі через кодуючу та декодуючу частини автоенкодера, обчислення та аналізу помилки реконструкції для виявлення аномалій. Нижче наведено блок-схему, що відображає алгоритм роботи запропонованого підходу, що лежить в основі ПЗ (рис. 2.1).

Крок 1. Початок. Починається процес виявлення аномалій за допомогою навченого автоенкодера в промисловій системі керування.

Система готується до обробки вхідних даних.

Крок 2. Ініціалізація системи підключення до БД. На цьому етапі відбувається встановлення з'єднання з базою даних, завантаження необхідних конфігураційних параметрів та підготовка системи до роботи. Перевіряється доступність бази даних та коректність збережених навчальних даних і моделі автоенкодера.

Крок 3. Завантаження навченої моделі автоенкодера.

Система завантажує попередньо навчену модель автоенкодера з файлової системи або бази даних. Модель була навчена на історичних даних нормального

функціонування промислової системи і готова до виявлення відхилень від звичайного режиму роботи.

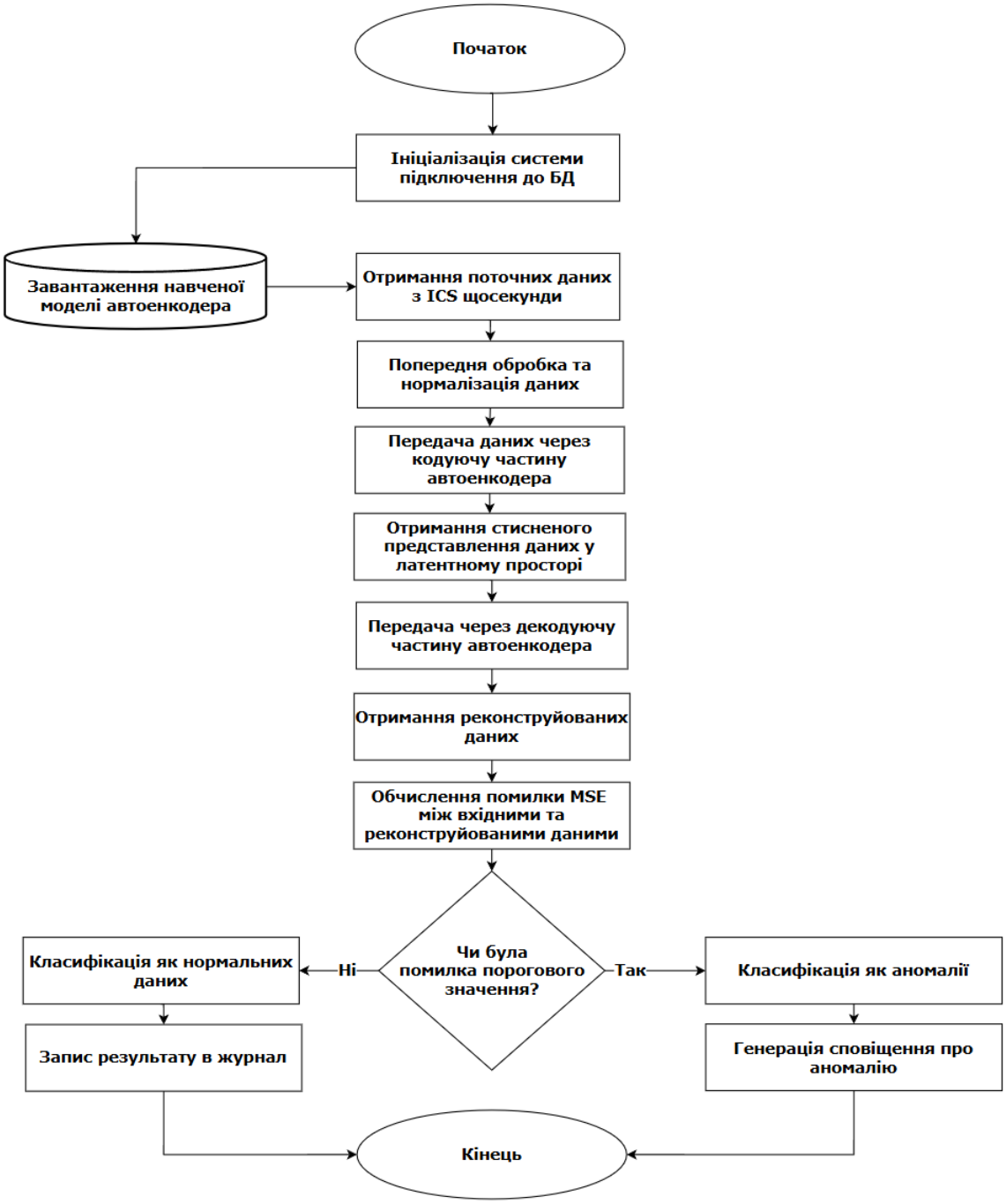


Рисунок 2.1 – Блок-схема алгоритму роботи запропонованого підходу, що лежить в основі програмного засобу

Крок 4. Отримання поточних даних з ICS щосекунди.

На цьому етапі відбувається збір телеметричних даних з промислової системи керування з частотою один раз на секунду. Збираються дані про

ключові технологічні параметри: температуру, тиск та рівень води. Це реальні дані від сенсорів та контролерів, які відображають поточний стан системи та потребують аналізу на наявність можливих аномалій.

Крок 5. Попередня обробка та нормалізація даних.

Отримані сирі дані проходять етап попередньої обробки, який включає фільтрацію шумів, усунення викидів та нормалізацію значень до діапазону, сумісного з навченою моделлю автоенкодера. Цей етап критично важливий для забезпечення коректної роботи нейронної мережі.

Крок 6. Передача даних через кодуючу частину автоенкодера.

Нормалізовані дані подаються на вхід кодуючої частини автоенкодера. Кодер перетворює тривимірний вектор вхідних параметрів (температура, тиск, рівень води) у стиснене представлення меншої розмірності, виділяючи найважливіші характеристики даних.

Крок 7. Отримання стисненого представлення даних у латентному просторі.

У результаті роботи кодера формується стиснене представлення вхідних даних у так званому латентному просторі. Це представлення містить ключову інформацію про вхідні параметри в компактній формі та служить основою для подальшої реконструкції.

Крок 8. Передача через декодуючу частину автоенкодера.

Стиснене представлення подається на вхід декодуючої частини автоенкодера, яка намагається відновити початкові дані з їх компактного представлення. Декодер використовує навчені ваги для реконструкції тривимірного вектора параметрів.

Крок 9. Отримання реконструйованих даних.

У результаті роботи декодера отримуються реконструйовані значення технологічних параметрів. Якість реконструкції залежить від того, наскільки вхідні дані відповідають нормальному режиму роботи, на якому навчався автоенкодер.

Крок 10. Обчислення помилки MSE між вхідними та реконструйованими даними.

На цьому етапі розраховується середньоквадратична похибка (Mean Squared Error, MSE) між початковими та реконструйованими значеннями параметрів. Високе значення MSE вказує на значні відмінності між вхідними та відновленими даними, що може свідчити про наявність аномалії.

Крок 11. Перевірка порогового значення та класифікація.

Розрахована помилка MSE порівнюється з попередньо встановленим пороговим значенням. Якщо MSE перевищує поріг, дані класифікуються як аномалія, інакше - як нормальні. Це рішення визначає подальші дії системи щодо реагування на виявлені відхилення.

Крок 12a. Класифікація як нормальних даних (MSE ≤ поріг).

Якщо розрахована помилка не перевищує порогове значення, поточний стан системи вважається нормальним. Інформація про нормальний стан записується у журнал системи для подальшого аналізу та звітності.

Крок 12b. Класифікація як аномалії (MSE > поріг).

Якщо помилка реконструкції перевищує встановлений поріг, система класифікує поточний стан як аномалію. Це може вказувати на кібератаку, технічну несправність або інший нештатний режим роботи промислової системи.

Крок 13a. Запис результату у журнал.

Для нормальних даних у базу даних записується інформація про поточний стан системи, включаючи часову мітку, значення технологічних параметрів, розраховану помилку MSE та статус "нормальний".

Крок 13b. Генерація сповіщення про аномалію.

При виявленні аномалії система генерує сповіщення для операторів або систем автоматичного реагування. Сповіщення містить детальну інформацію про характер виявленої аномалії та рекомендації щодо дій.

Крок 14. Запис результату та деталей аномалії у журнал БД.

Детальна інформація про виявлену аномалію записується у базу даних, включаючи часову мітку, значення всіх технологічних параметрів, величину помилки MSE, статус "аномалія" та додаткові деталі для подальшого розслідування інциденту.

Крок 15. Кінець.

Процес виявлення аномалій для поточного набору даних завершено. Система готова до аналізу наступної порції даних з промислової системи керування через одну секунду, забезпечуючи безперервний моніторинг у режимі реального часу.

Запропонований підхід також враховує особливості промислових систем керування, такі як обмежена обчислювальна потужність контролерів, необхідність роботи в режимі реального часу та критичність хибних спрацьовувань. Для забезпечення надійної роботи системи виявлення аномалій в таких умовах було оптимізовано архітектуру автоенкодера та алгоритми обробки даних.

Таким чином, розроблений алгоритм забезпечує комплексний підхід до виявлення аномалій у промислових системах керування, поєднуючи переваги машинного навчання з практичними вимогами промислового середовища, включаючи швидкодію, надійність та мінімізацію хибних спрацьовувань.

2.2 Проєктування архітектури навчання автоенкодера для виявлення аномалій на ICS та взаємодія із базою даних

Наведений у попередньому підрозділі алгоритм роботи запропонованого підходу базується на використанні попередньо навченого автоенкодера, який здатен ефективно виявляти аномалії в промислових системах керування шляхом аналізу помилки реконструкції технологічних параметрів. Однак для забезпечення високої якості виявлення аномалій критично важливим є процес навчання самого автоенкодера, який повинен бути спеціально адаптований до специфіки конкретної промислової системи та характеристик її нормального

функціонування. Враховуючи складність отримання репрезентативних навчальних даних з реальних промислових систем та необхідність забезпечення контрольованого процесу навчання без присутності аномальних прикладів, було прийнято рішення про розробку окремого модуля навчання автоенкодера з використанням згенерованих навчальних даних.

У рамках розробки програмного засобу для виявлення атак на промислові системи керування з використанням автоенкодерів було спроектовано модуль обробки навчального датасету та навчання автоенкодера. Цей модуль відіграє ключову роль у загальній системі, оскільки саме якість навчання автоенкодера визначає ефективність виявлення аномалій в подальшому. Для побудови ефективної системи виявлення аномалій на промислових системах керування було обрано підхід, заснований на автоенкодері – спеціальному типі нейронної мережі, що навчається відтворювати вхідні дані на виході, проходячи через шар з меншою розмірністю (латентний простір). Цей підхід дозволяє виявляти аномалії без необхідності мати попередні приклади атак, що особливо важливо для промислових середовищ, де набори даних про атаки обмежені або відсутні.

Навчання автоенкодера включала кілька ключових етапів: генерацію навчального датасету, обробку та нормалізацію даних, проектування архітектури автоенкодера, налаштування параметрів навчання та власне процес навчання. Для навчання автоенкодера було прийнято рішення використовувати згенерований датасет, що моделює нормальну роботу промислової системи керування. Це обумовлено складністю отримання реальних даних з промислових систем та необхідністю мати контрольований набір даних без аномалій для коректного навчання автоенкодера.

Згенерований датасет містив три ключові параметри, що характеризують роботу типової промислової системи керування: тиск у трубах (у кПа), рівень рідини (у відсотках від максимального об'єму) та температура (у градусах Цельсія). Для кожного параметра було згенеровано по 1000 значень, що відповідають нормальним режимам роботи системи. Обмеження обсягу

датасету до 1000 записів для кожного параметра було свідомим рішенням, оскільки для моделі з відносно невеликою кількістю вхідних ознак використання більшого обсягу даних могло призвести до перенавчання та нестабільної роботи моделі.

Діапазони значень були визначені відповідно до типових технологічних норм для промислових систем: тиск у трубах 100-500 кПа, рівень рідини 20-80% та температура 50-90°C. Згенеровані дані включали природні флуктуації, характерні для реальних промислових процесів, але без аномальних значень, що могли б вказувати на збої або атаки. Перед подачею на вхід автоенкодера дані пройшли етап попередньої обробки та нормалізації, що є критично важливим для ефективного навчання нейронних мереж.

Було застосовано перевірку та обробку пропущених значень, хоча згенерований датасет не мав пропусків, у розробленому модулі було реалізовано механізм виявлення та заповнення пропущених значень для забезпечення роботи з реальними даними в майбутньому. Нормалізація значень до діапазону [0, 1] була необхідна для приведення різних за масштабом параметрів до єдиного діапазону, що значно покращує процес навчання нейронної мережі та запобігає домінуванню певних ознак через їх більший числовий масштаб. Математично процес нормалізації можна представити формулою:

$$X_n = \frac{X - X_{min}}{(X_{max} - X_{min})}, \quad (2.1)$$

де X– початкове значення параметра;

X_{min} – мінімальне значення в наборі даних;

X_{max}– функція навчання;

На основі аналізу специфіки задачі та характеристик вхідних даних було спроектовано архітектуру автоенкодера, що складається з кодуючої та декодуючої частин, реалізованої з використанням бібліотеки TensorFlow.

Кодуюча частина автоенкодера відповідає за стиснення вхідних даних до меншої розмірності, а декодуюча частина – за відновлення початкових даних з

їх стисненого представлення. Структура автоенкодера включає вхідний шар з 3 нейронами, кодуючу частину з двома прихованими шарами по 16 та 8 нейронів відповідно з функцією активації ReLU, латентний простір з 4 нейронами, декодуючу частину з прихованими шарами по 8 та 16 нейронів з функцією активації ReLU та вихідний шар з 3 нейронами з лінійною функцією активації.

Перед початком процесу навчання автоенкодера було налаштовано наступні параметри: функція втрат середньоквадратична похибка (MSE), оптимізатор Adam, кількість епох навчання 500, розмір міні-батчу 32 та швидкість навчання 0.001. Ці параметри були підібрані експериментально з урахуванням специфіки задачі та характеристик навчального датасету. Навчання автоенкодера відбувалося шляхом мінімізації помилки реконструкції між вхідними та вихідними даними на наборі виключно нормальних даних, що дозволяє йому вивчити нормальну поведінку системи і в подальшому виявляти відхилення від неї.

Під час навчання відстежувалася динаміка зміни функції втрат, що дозволяло контролювати процес навчання та запобігати перенавчанню за допомогою механізму ранньої зупинки, який призупиняв навчання, якщо протягом певної кількості епох не спостерігалося покращення функції втрат на валідаційному наборі даних. Успішно навчена модель автоенкодера зберігається в базі даних SQLite разом з усіма супутніми метаданими для подальшого використання в процесі виявлення аномалій у режимі реального часу. Процес навчання автоенкодера представлено у вигляді блок–схеми (рис. 2.2).

Крок 1. Початок.

Починається процес навчання автоенкодера для виявлення аномалій у промислових системах керування (ICS). Система готується до створення навченої моделі, яка буде здатна розпізнавати відхилення від нормальної поведінки технологічних параметрів.

Крок 2. Генерація навчальних даних.

На цьому етапі відбувається створення або завантаження набору навчальних даних, які містять показники температури, тиску та рівня води в нормальному режимі роботи промислової системи. Ці дані є еталонними прикладами штатного функціонування обладнання.

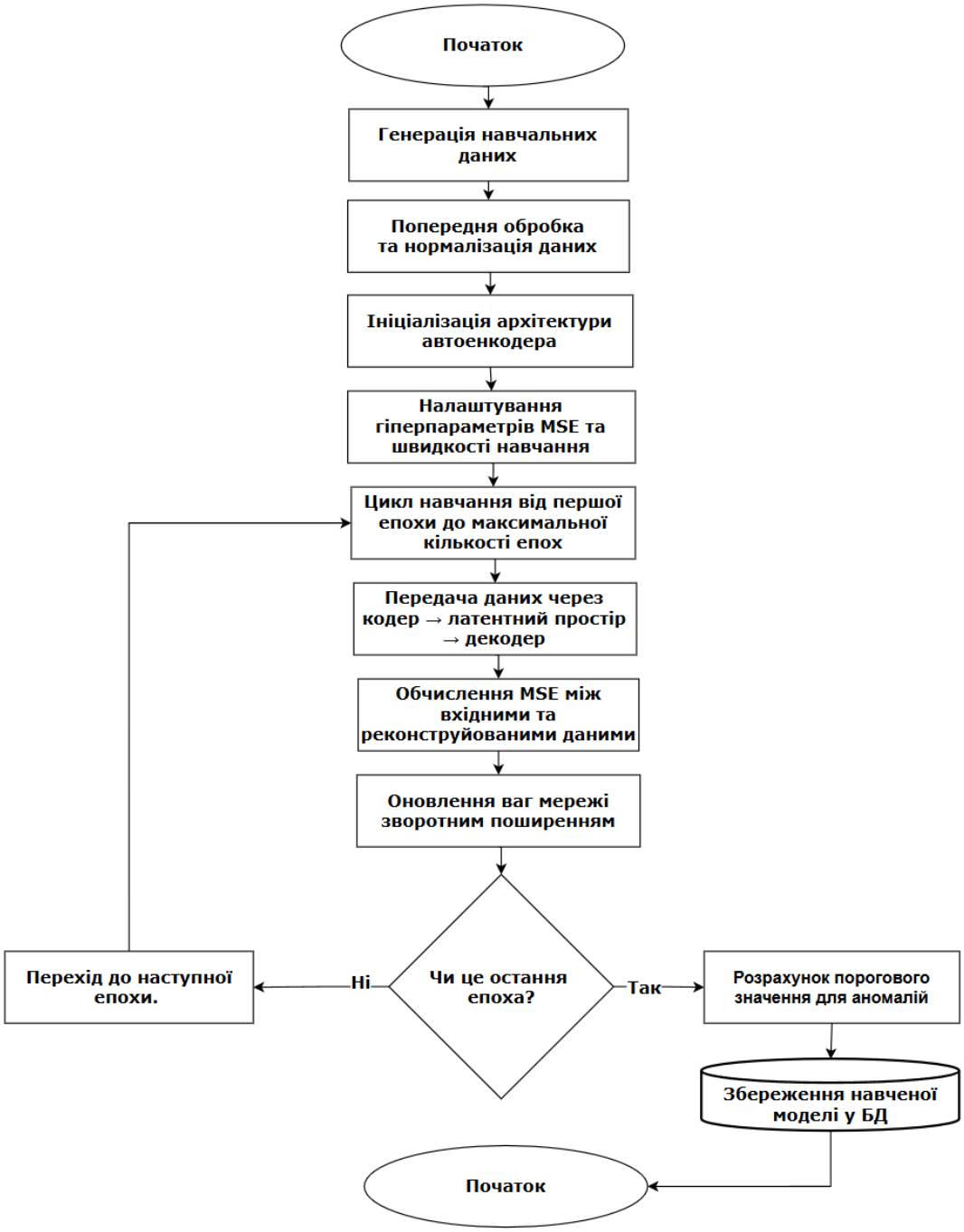


Рисунок 2.2 – Блок-схеми алгоритму навчання автоенкодера

Крок 3. Попередня обробка та нормалізація даних.

Завантажені навчальні дані проходять етап попередньої обробки, який включає очищення від викидів, усунення пропущених значень та нормалізацію параметрів до діапазону [0,1]. Це забезпечує стабільність процесу навчання та покращує збіжність алгоритму.

Крок 4. Ініціалізація архітектури автоенкодера.

Система створює структуру нейронної мережі автоенкодера, яка складається з кодуючої частини (encoder), латентного простору та декодуючої частини (decoder). Архітектура оптимізована для обробки трьохвимірних векторів вхідних технологічних параметрів.

Крок 5. Налаштування гіперпараметрів MSE та швидкості навчання.

На цьому етапі встановлюються ключові параметри процесу навчання: функція втрат MSE (середньоквадратична похибка), швидкість навчання (learning rate), розмір пакетів даних (batch size) та максимальна кількість епох навчання.

Крок 6. Цикл навчання від першої епохи до максимальної кількості епох.

Починається ітеративний процес навчання автоенкодера. Кожна епоха представляє повний прохід через весь навчальний датасет. Система встановлює лічильник поточної епохи та готується до циклічного процесу оптимізації ваг мережі.

Крок 7. Передача даних через кодер → латентний простір → декодер.

Навчальні дані подаються на вхід автоенкодера та послідовно проходять через кодуючу частину, де відбувається стиснення інформації до латентного представлення, а потім через декодуючу частину, яка намагається відновити початкові дані.

Крок 8. Обчислення MSE між вхідними та реконструйованими даними.

Система розраховує середньоквадратичну похибку між початковими значеннями технологічних параметрів та їх реконструйованими версіями на виході автоенкодера. Цей показник характеризує якість відтворення нормальних даних моделлю.

Крок 9. Оновлення ваг мережі зворотним поширенням.

На основі обчисленої похибки відбувається коригування ваг нейронної мережі методом зворотного поширення помилки та градієнтного спуску. Це дозволяє автоенкодеру поступово покращувати якість реконструкції нормальних даних.

Крок 10. Перевірка: чи це остання епоха?

Система перевіряє, чи досягнуто максимальну кількість епох навчання або чи виконані критерії дострокового зупину (early stopping). Якщо навчання має продовжуватися, лічильник епох збільшується на одиницю та процес повертається до кроку 6.

Крок 11. Перехід до наступної епохи.

У випадку, коли навчання не завершено, система збільшує номер поточної епохи на одиницю та повертається до циклу навчання для подальшої оптимізації параметрів автоенкодера на тому ж навчальному датасеті.

Крок 12. Розрахунок порогового значення для аномалій.

Після завершення процесу навчання система аналізує розподіл помилок реконструкції на навчальних даних та визначає оптимальне порогове значення, перевищення якого буде сигналізувати про наявність аномалії в технологічних параметрах.

Крок 13. Збереження навченої моделі у БД.

Навчений автоенкодер разом з розрахованим пороговим значенням та метаданими процесу навчання зберігається в базі даних SQLite для подальшого використання в системі виявлення аномалій у режимі реального часу.

Крок 14. Кінець.

Процес навчання автоенкодера завершено. Система має готову до використання модель, яка здатна ефективно виявляти аномалії в промислових системах керування на основі аналізу технологічних параметрів температури, тиску та рівня води.

Запропонований алгоритм навчання враховує специфічні вимоги промислових систем керування, такі як необхідність високої точності виявлення аномалій, мінімізація хибних спрацьовувань та ефективність обчислень. Для забезпечення стабільної роботи в промисловому середовищі було оптимізовано архітектуру автоенкодера та параметри процесу навчання.

Критично важливим етапом у реалізації системи виявлення аномалій є організація ефективного та безпечного зберігання навчених моделей автоенкодера разом із супутніми метаданими. Для цієї мети було обрано базу даних SQLite як оптимальне рішення для локального зберігання даних, що поєднує в собі простоту розгортання, надійність роботи та достатню функціональність для потреб розробленої системи. SQLite є вбудованою базою даних, яка не вимагає окремого серверного процесу та дозволяє зберігати всі дані в єдиному файлі, що значно спрощує управління та резервне копіювання інформації.

Архітектура бази даних була спроектована з урахуванням специфічних потреб системи моніторингу промислових процесів та включає три основні таблиці для зберігання різних типів інформації. Таблиця TRAINING_DATA призначена для збереження навчальних даних, що використовувалися для тренування автоенкодера, та містить поля для унікального ідентифікатора запису, значень температури, тиску та рівня води. Ця таблиця забезпечує можливість відтворення процесу навчання та аналізу використаних навчальних даних у разі необхідності подальшого вдосконалення моделі або діагностики її роботи. Нижче наведено ER-модель задіяної в ПЗ бази даних (рис. 2.3).

Таблиця LOG_DATA відіграє ключову роль у забезпеченні трасованості роботи системи та містить детальну інформацію про кожен акт моніторингу технологічних параметрів. Структура цієї таблиці включає унікальний ідентифікатор, мітку часу операції, поточні значення контрольованих параметрів (температура, тиск, рівень води), розраховане значення середньоквадратичної похибки реконструкції та статус виявлення аномалії. Така

організація даних дозволяє проводити ретроспективний аналіз роботи системи, виявляти тенденції в поведінці технологічних параметрів та оптимізувати налаштування системи виявлення аномалій.

Додатково було створено таблицю SYSTEM_PARAMETERS для зберігання конфігураційних параметрів системи, що включає назви параметрів, їх поточні значення, мінімальні та максимальні порогові значення, а також інформацію про час останнього оновлення.

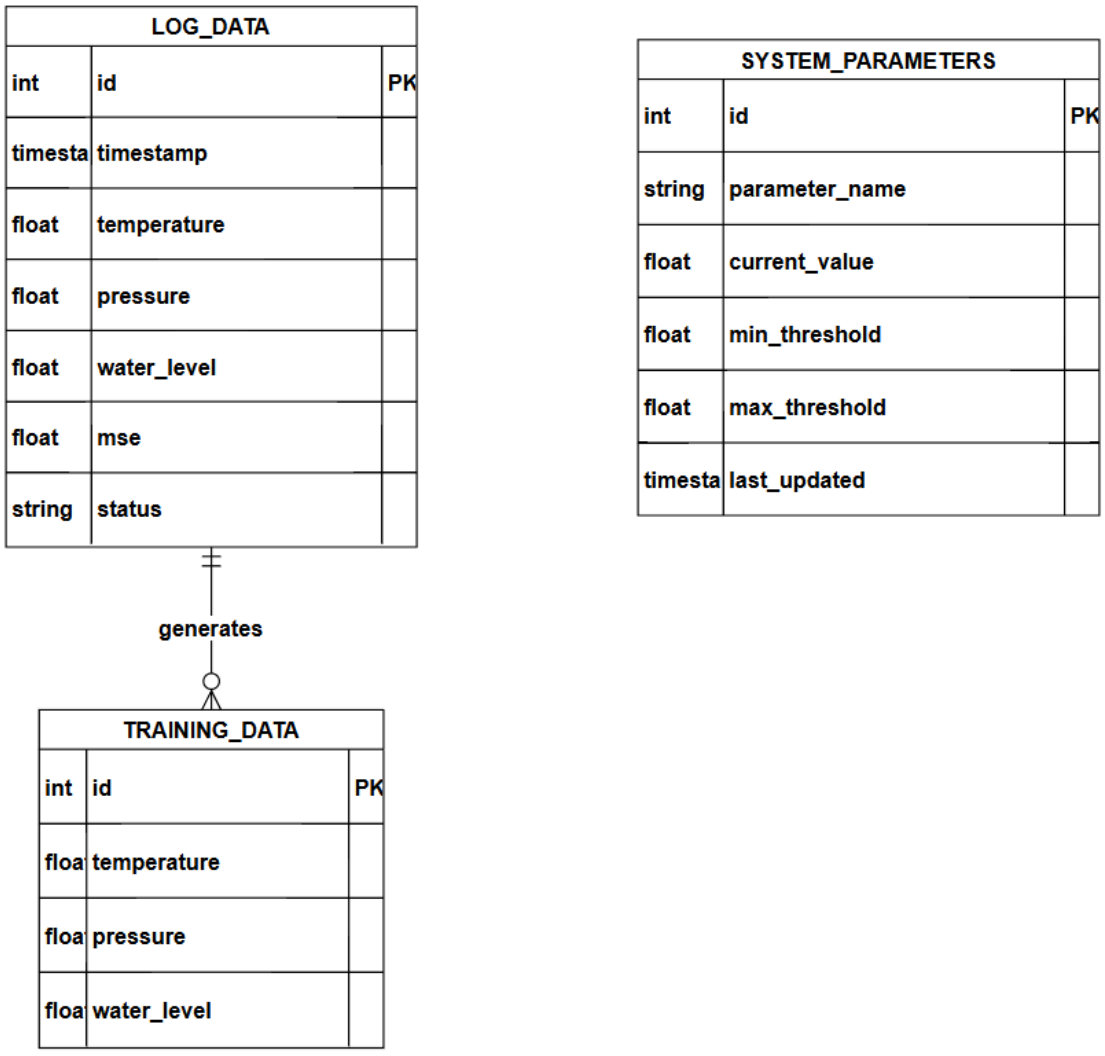


Рисунок 2.3 – ER-модель задіяної в програмному засобі бази даних

Це забезпечує гнучкість налаштування системи та можливість динамічного коригування параметрів роботи без необхідності перекомпіляції програмного коду. Особливу увагу було приділено безпеці зберігання даних, оскільки

інформація про промислові системи може бути конфіденційною. Для захисту бази даних SQLite використано шифрування на основі бібліотеки SQLCipher, яка реалізує AES-256 у режимі CBC з використанням PBKDF2 для генерації ключа з пароля. SQLite як вбудоване рішення зберігає всю інформацію в одному файлі, що дозволяє ефективно контролювати його цілісність і запобігати сторонньому втручанню. Такий підхід забезпечує вищий рівень захисту в порівнянні з серверними СУБД, оскільки відсутність відкритого мережевого інтерфейсу зменшує площу потенційної атаки.

Інтеграція бази даних з основною системою виявлення аномалій здійснюється через легкий програмний інтерфейс (API), що обробляє запити на зчитування й запис безпосередньо в SQLite. Це дозволяє отримувати й зберігати дані в режимі реального часу без істотного навантаження на систему. Така архітектура забезпечує стабільну роботу в умовах промислового середовища, зберігаючи критично важливу інформацію в захищеному вигляді й забезпечуючи можливість швидкого відновлення у разі збою.

2.3 Проєктування загальної архітектури програмного засобу

У процесі проєктування програмного засобу для виявлення атак на промислові системи керування (ICS) з використанням автоенкодерів важливо розробити таку архітектуру, яка забезпечуватиме ефективне виконання всіх необхідних функцій, масштабованість, а також зручність використання кінцевими користувачами. Загальна архітектура програмного засобу повинна враховувати особливості роботи з даними ICS, специфіку функціонування автоенкодерів для виявлення аномалій, а також вимоги до інтерфейсу та інтеграції з існуючими системами моніторингу промислових об'єктів.

Розроблена архітектура програмного засобу базується на модульному підході, що забезпечує гнучкість у розробці, тестуванні та подальшому масштабуванні системи. Основними компонентами архітектури є: модуль збору та попередньої обробки даних, модуль навчання автоенкодера, модуль

виявлення аномалій, модуль візуалізації результатів та інтерфейс користувача. Кожен із зазначених модулів виконує специфічні функції та взаємодіє з іншими компонентами системи через визначені інтерфейси.

Модуль попередньої обробки даних відповідає за отримання даних та нормалізацію, фільтрацію та перетворення даних у формат, придатний для подальшої обробки автоенкодером. Модуль навчання автоенкодера реалізує функціонал для тренування нейромережевої моделі на підготовлених «нормальних» даних. Він включає в себе компоненти налаштування гіперпараметрів, а також контролю процесу навчання та валідації моделі. Він взаємодіє з модулем попередньої обробки даних для отримання тренувальних наборів та з модулем виявлення аномалій для передачі навченої моделі.

Модуль виявлення аномалій є ключовим компонентом системи, який використовує навчену модель автоенкодера для аналізу поточних даних ICS та ідентифікації потенційних атак або аномальної поведінки. Цей модуль обчислює похибку реконструкції для кожного вхідного вектора даних та порівнює її з визначеними порогами для прийняття рішення щодо наявності аномалії. Також модуль включає компоненти для обчислення метрик якості роботи системи виявлення (precision, recall, F1-score тощо). Інтерфейс користувача надає зручні інструменти для налаштування параметрів системи, перегляду результатів аналізу, керування процесами та моніторингу стану промислової системи. Архітектура системи також передбачає наявність модуля зберігання даних, який забезпечує ефективне збереження як вхідних даних з ICS, так і результатів аналізу, параметрів моделей та журналів подій.

Важливим аспектом архітектури є забезпечення масштабованості системи для роботи з різними типами та обсягами даних ICS. Для цього передбачена можливість горизонтального масштабування компонентів обробки даних та аналізу, а також використання розподілених обчислень для обробки великих обсягів інформації в режимі реального часу. Нижче наведена взаємодія компонентів архітектури розробленого програмного засобу (рис. 2.4).

Крок 1. Початок. Процес функціонування програмного засобу запускається.

Крок 2. Джерела даних ICS. Система отримує дані з різноманітних промислових систем керування (ICS) через підтримувані протоколи зв'язку.

Крок 3. Модуль попередньої обробки. Отримані дані проходять попередню обробку, що включає фільтрацію, перетворення у формат, придатний для аналізу.

Крок 4. Модуль навчання автоенкодера. На підготовлених даних здійснюється навчання нейромережевої моделі автоенкодера для формування образу нормальної поведінки системи.

Крок 5. Модуль збереження моделей. Навчені моделі автоенкодерів зберігаються для подальшого використання в системі виявлення аномалій.

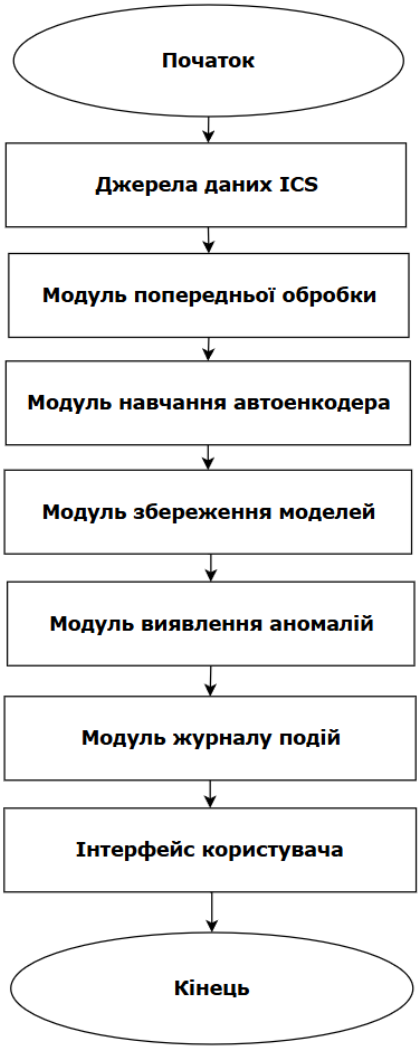


Рисунок 2.4 – Схема взаємодії компонентів архітектури розробленого ПЗ

Крок 6. Модуль виявлення аномалій. Збережені моделі використовуються для аналізу поточних даних з ICS та ідентифікації відхилень від нормальної поведінки, що можуть свідчити про атаки.

Крок 7. Модуль журналу подій. При виявленні аномалій генеруються відповідні сповіщення до журналу подій з описом характеру та потенційної небезпеки виявлених відхилень.

Крок 8. Інтерфейс користувача. Інформація про виявлені аномалії та стан системи відображається користувачу через графічний інтерфейс для прийняття рішень щодо реагування.

Крок 9. Кінець. Процес завершується, але система продовжує працювати циклічно, постійно аналізуючи нові дані.

Ця покрокова схема взаємодії компонентів архітектури програмного засобу демонструє повний цикл обробки даних від їх отримання з ICS до відображення результатів аналізу користувачу. Архітектура забезпечує послідовну обробку інформації через взаємопов'язані модулі, кожен з яких виконує специфічні функції в загальному процесі виявлення атак на ICS з використанням автоенкодерів. Нижче наведено взаємодію користувача з функціоналом розробленого програмного засобу (на рис. 2.5).

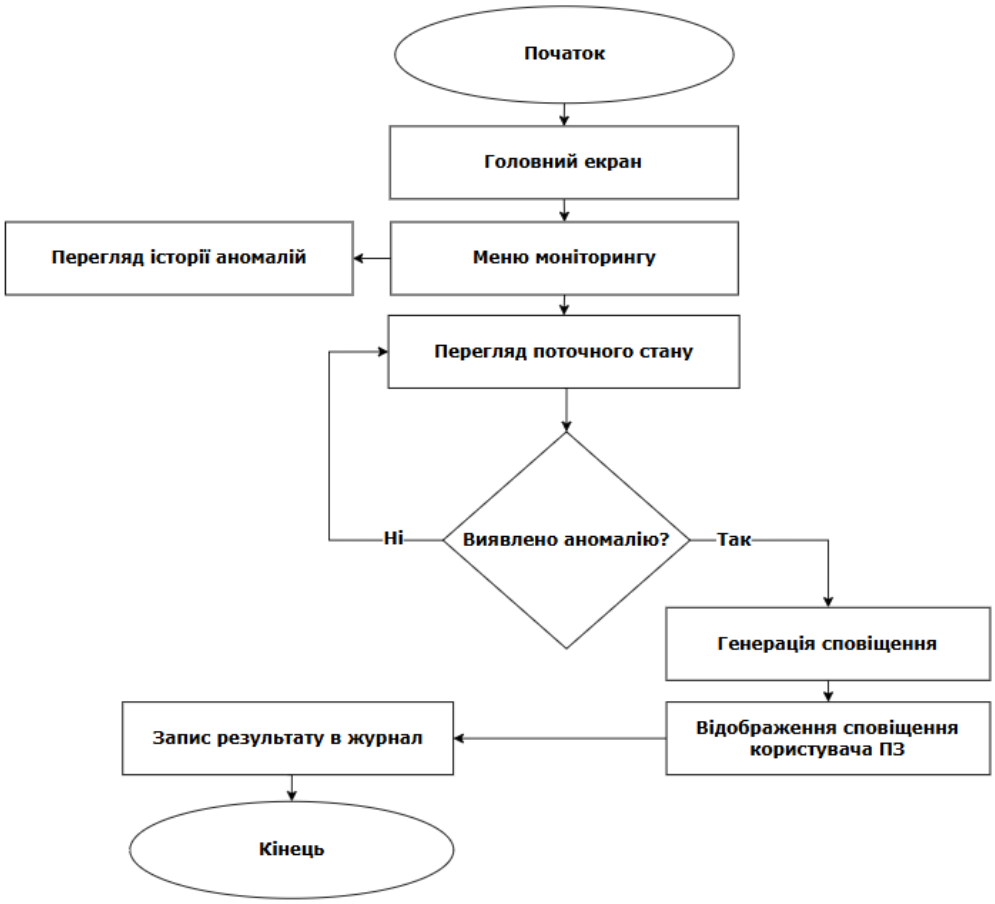


Рисунок 2.5 – Схема взаємодії користувача з функціоналом розробленого ПЗ

Крок 1. Початок.

Користувач починає взаємодіяти з програмним засобом.

Крок 2. Головний екран.

Користувач отримує доступ до головного екрану системи, де представлені основні функції та елементи управління.

Крок 3. Меню моніторингу.

Користувач переходить до меню моніторингу, що надає інструменти для спостереження за станом промислових систем.

Крок 4. Перегляд історії аномалій.

За необхідності користувач може переглянути історію раніше виявлених аномалій, щоб проаналізувати тенденції та патерни атак.

Крок 5. Перегляд поточного стану.

Користувач переходить до інтерфейсу перегляду поточного стану системи, де відображаються актуальні дані з ICS та результати їх аналізу.

Крок 6. Виявлено аномалію?

Система автоматично аналізує дані та визначає наявність аномалій. На цьому етапі відбувається розгалуження процесу залежно від результату аналізу.

Крок 7а. (Відповідь "Так") Генерація сповіщення.

При виявленні аномалії система генерує сповіщення з детальною інформацією про характер виявленого відхилення.

Крок 8а. Відображення сповіщення користувачу ПЗ.

Сформоване сповіщення відображається користувачу через інтерфейс програмного засобу для прийняття рішення щодо реагування.

Крок 7б. (Відповідь "Ні")

Повернення до перегляду поточного стану. Якщо аномалії не виявлено, система продовжує моніторинг поточного стану ICS.

Крок 9. Запис результату в журнал.

Результати аналізу (як виявлені аномалії, так і їх відсутність) записуються в журнал подій для подальшого аудиту та аналізу.

Крок 10. Кінець.

Процес взаємодії користувача з функціоналом моніторингу завершується.

Ця покрокова схема взаємодії користувача з програмним засобом демонструє логічну послідовність дій при використанні функціоналу моніторингу та виявлення аномалій. Схема наочно відображає циклічний характер процесу моніторингу з можливістю реагування на виявлені аномалії та ведення журналу подій для забезпечення аудиту та аналізу безпеки промислових систем керування.

Таким чином, розроблена архітектура програмного засобу для виявлення атак на промислові системи керування ICS з використанням автоенкодерів для виявлення аномалій забезпечує комплексне рішення для моніторингу безпеки промислових об'єктів. Модульний підхід до проєктування архітектури дозволяє забезпечити гнучкість, масштабованість та надійність системи, а також спрощує процес її розробки, тестування та подальшої підтримки.

2.4 Висновки до розділу 2

У цьому розділі запропоновано підхід до виявлення аномалій на промислових системах керування (ICS) з використанням автоенкодерів. Обґрунтовано перевагу автоенкодерів перед традиційними сигнатурними методами, які обмежені виявленням лише відомих атак та не забезпечують належного захисту в умовах постійного розвитку нових векторів загроз.

Запропонований метод дозволяє виявляти невідомі "zero-day" атаки завдяки архітектурі автоенкодера, що складається з кодера (стиснення даних) та декодера (відновлення даних). Принцип роботи базується на тому, що система, навчена на нормальних даних, демонструватиме високу помилку реконструкції при обробці аномальних даних. Алгоритм включає отримання даних з ICS, їх попередню обробку, проходження через автоенкодер та аналіз помилки реконструкції. Розроблена модульна архітектура програмного засобу забезпечує гнучкість системи та охоплює повний цикл обробки даних в реальному часі з можливістю моніторингу та ведення журналу подій.

Наступним етапом є практична реалізація програмного засобу та його експериментальне дослідження з оцінкою точності виявлення аномалій, кількості хибних спрацьовувань та швидкодії.

3 РЕАЛІЗАЦІЯ ПЗ ДЛЯ ВИЯВЛЕННЯ АТАК НА ПРОМИСЛОВІ СИСТЕМИ КЕРУВАННЯ (ICS) З ВИКОРИСТАННЯМ АВТОЕНКОДЕРІВ

Третій розділ присвячений практичній реалізації програмного засобу для виявлення кіберзагроз у промислових системах керування на основі розроблених у попередньому розділі концептуальних рішень. У розділі обґрунтовано вибір технологічного стеку, включаючи мову програмування Python та спеціалізовані бібліотеки TensorFlow, Scikit-learn і Tkinter, що забезпечують оптимальне поєднання продуктивності та функціональності для задач машинного навчання. Детально описано архітектуру програмного засобу з модульною структурою, що включає компоненти обробки даних, навчання автоенкодера та користувацького інтерфейсу. Проведено комплексне тестування ефективності системи з оцінкою точності виявлення аномалій, аналізом помилок та швидкодії роботи у режимі реального часу. Розділ містить результати експериментальної перевірки на модельованих даних ICS, що підтверджують здатність системи виявляти як поодинокі викиди параметрів, так і складні багатофакторні аномалії, характерні для цілеспрямованих кібератак на критичну інфраструктуру.

3.1 Обґрунтування вибору мови програмування та інструментів розробки

Розробка програмного засобу для виявлення атак на промислові системи керування (ICS) з використанням автоенкодерів потребує вибору мови програмування та інструментів, які забезпечують простоту реалізації, наявність потужних бібліотек для машинного навчання, а також можливість створення графічного інтерфейсу користувача.

У процесі вибору розглядалися такі популярні мови програмування, як Python, C++, Java та C#. Основними критеріями оцінки були: підтримка глибокого навчання, зручність реалізації, наявність відповідних бібліотек, а також можливість швидкого створення прототипу.

Python є безумовним лідером у сфері машинного навчання. Завдяки таким бібліотекам, як TensorFlow, Keras, PyTorch, а також наявності високорівневих API, Python дозволяє швидко реалізовувати та тестувати моделі глибокого навчання, зокрема автоенкодери. Крім того, Python підтримує створення GUI-додатків за допомогою бібліотек Tkinter, PyQt або Kivy, що є зручним для реалізації користувацького інтерфейсу без необхідності застосування складних фреймворків.

C++ та C# забезпечують високу продуктивність, однак розробка моделей машинного навчання на цих мовах потребує значно більше зусиль, а також використання додаткових бібліотек або обгорток. Java має певну підтримку глибокого навчання (наприклад, через Deeplearning4j), проте поступається Python у гнучкості та кількості готових рішень.

У процесі розробки програмного засобу для виявлення атак на промислові системи керування ICS з використанням автоенкодерів, дуже важливо правильно обрати мову програмування. У таблиці нижче наведено порівняння найпопулярніших мов програмування, які використовуються в цій галузі. (табл.3.1).

Таблиця 3.1 – Порівняння відомих фреймворків для машинного навчання

Характеристика	Python	C++	Java	C#
Машинне навчання	+	+/–	+/–	+/–
Простота реалізації	+	–	+/–	+
Наявність бібліотек (ML)	+++	+	+	+/–
Швидкість розробки	+++	–	++	++
Побудова GUI	++	+	++	++

На основі аналізу було обрано Python як оптимальну мову програмування для реалізації програмного засобу через такі переваги:

– ..широкий вибір бібліотек для реалізації автоенкодерів, таких як TensorFlow, Keras та scikit-learn;

– ..швидка розробка прототипів завдяки простому синтаксису;

– ..можливість створення графічного інтерфейсу з використанням бібліотеки Tkinter, яка входить до стандартної бібліотеки Python;

– ..активна підтримка спільноти та велика кількість готових прикладів, що значно полегшує розробку.

Таким чином, у підсумку Python було обрано не лише через наявність широкого спектра бібліотек, а й завдяки поєднанню гнучкості, легкості розробки, зручності створення прототипів і доступності рішень для GUI, що особливо важливо в межах обмежених ресурсів навчального або дослідницького проєкту.

Оскільки основою системи є побудова та навчання автоенкодера, постає питання вибору відповідного фреймворку глибокого навчання. Серед основних варіантів розглянуто TensorFlow, PyTorch, Keras, MXNet та JAX. TensorFlow має одну з найширших екосистем, підтримує масштабування, оптимізований для використання GPU, та дозволяє реалізовувати нейронні мережі на різних рівнях абстракції. PyTorch, своєю чергою, є більш гнучким у дослідницьких задачах, має інтуїтивно зрозумілу імперативну модель, що спрощує відлагодження моделей та їх візуалізацію. Keras, як високорівнева обгортка над TensorFlow, дає змогу створювати моделі дуже швидко, однак обмежений у гнучкості. MXNet та JAX мають вузьке коло використання, менше прикладів і менш розвинену спільноту, що може ускладнити реалізацію. Нижче наведено порівняння популярних фреймворків з підтримкою глибокого навчання автоенкодерів (табл. 3.2).

Таблиця 3.2 – Порівняння відомих фреймворків для роботи із автоенкодерами

Фреймворк	Мова	Підтримка автоенкодерів	Простота використання	Документація
TensorFlow	Python	+++	++	+++
Keras	Python	+++	+++	++

PyTorch	Python	+++	++	+++
ML.NET	C#	+	+	+++
Deeplearning4j	Java	+	+/–	++

Для реалізації машинного навчання у даному проєкті обрано фреймворк TensorFlow, оскільки він:

- підтримує побудову та навчання автоенкодерів як на низькому, так і на високому рівні (через Keras API);
- забезпечує високу продуктивність завдяки використанню GPU;
- має хорошу документацію та активну підтримку.

З огляду на вищезазначене, TensorFlow був обраний як оптимальний варіант, він поєднує продуктивність, гнучкість, підтримку GPU та має інтеграцію з Keras, що дозволяє реалізувати автоенкодери з мінімальними зусиллями як на початковому, так і на просунутому рівні.

Tkinter, як стандартна бібліотека Python, дозволяє створювати GUI без додаткових залежностей, що особливо важливо в умовах обмежених ресурсів. У той час як PyQt та Kivy мають ширший функціонал і можливості кастомізації, їх використання передбачає складнішу інтеграцію та вищий поріг входу, що не завжди виправдано для задач з простою структурою інтерфейсу.

Для створення графічного інтерфейсу користувача обрано Tkinter – легкий та простий інструмент для побудови GUI у Python. Його перевагами є:

- мінімальні залежності;
- можливість створення простого та інтуїтивно зрозумілого інтерфейсу;

Вибір зв’язки Python та TensorFlow обумовлений поєднанням ряду ключових переваг: наявністю потужних бібліотек, зручністю синтаксису, активною спільнотою, високою продуктивністю при використанні GPU, а також відносною простотою реалізації архітектур нейронних мереж, зокрема автоенкодерів. Такий вибір дозволяє сконцентруватися безпосередньо на логіці моделі, а не на технічних деталях реалізації, що є критично важливим для ефективної побудови системи виявлення атак на ICS. Tkinter було обрано як основну бібліотеку для створення графічного інтерфейсу користувача. Це

рішення зумовлено її простотою, наявністю у стандартній бібліотеці Python, відсутністю потреби в додаткових установках, а також достатньою функціональністю для реалізації базових інтерфейсних елементів, необхідних у межах даного проєкту.

Отже, для реалізації програмного засобу виявлення атак на ICS з використанням автоенкодерів найдоцільнішим є вибір стеку технологій: Python, TensorFlow, Tkinter.

3.2 Опис функціональних модулів програмної реалізації розробленого програмного засобу

Для реалізації запропонованого підходу до виявлення аномалій на промисловій системі керування ICS з використанням автоенкодерів у даних технологічного процесу, було розроблено програмний засіб мовою програмування Python з використанням бібліотек TensorFlow, NumPy, sklearn та засобів побудови графічного інтерфейсу tkinter. Програмний продукт призначений для моделювання нормальної поведінки системи на основі навчання автоенкодера, аналізу вхідних даних у режимі реального часу, визначення рівня схожості з нормальним станом та ідентифікації потенційних аномалій. Фрагмент нижче відповідає за запуск графічного інтерфейсу користувача з основного файлу.

```
# main.py
from gui import MonitoringApp
import tkinter as tk
def main():
    root = tk.Tk()
    app = MonitoringApp(root)
    root.mainloop()
if __name__ == "__main__":
    main()
```

Даний фрагмент коду вище відповідає за запуск графічного інтерфейсу користувача. Модуль main.py ініціалізує головне вікно застосунку за допомогою бібліотеки tkinter, створює екземпляр класу MonitoringApp, що реалізує інтерфейс моніторингу, та запускає основний цикл програми. Цей підхід

дозволяє користувачеві взаємодіяти із системою у зручному візуальному середовищі. Фрагмент нижче імпортує необхідні бібліотеки для побудови нейронної мережі, обробки даних та збереження моделі.

```
# model_handler.py (фрагмент 1)
import numpy as np
import os
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.callbacks import EarlyStopping
```

Підключаються бібліотеки для роботи з масивами, масштабування даних, побудови глибинної моделі та забезпечення зупинки тренування при стабілізації помилки. Фрагмент нижче задає шляхи до файлів, де зберігаються модель автоенкодера, параметри масштабування та поріг для виявлення аномалій.

```
MODEL_FILE = "autoencoder_model.keras"
SCALER_FILE = "scaler.npy"
THRESHOLD_FILE = "threshold.npy"
```

Ці константи використовуються для збереження та завантаження основних елементів системи: натренованої моделі, масштабувальника та значення порогу середньоквадратичної помилки (MSE), що визначає межу між нормальними та аномальними даними. Фрагмент нижче оголошує клас, відповідальний за обробку моделі автоенкодера: її створення, навчання та застосування.

```
class AutoencoderHandler:
    def __init__(self):
        self.model = None
        self.scaler = MinMaxScaler()
        self.threshold = None
        self._load_or_train_model()
```

У конструкторі створюється об’єкт масштабувальника та викликається метод, який або завантажує збережену модель, або тренує її заново у разі відсутності файлів. Фрагмент нижче генерує синтетичний набір нормальних даних, який використовуватиметься для навчання автоенкодера.

```
def _create_training_data_array(self, length):
    data = []   for _ in range(length):
        temperature = np.random.uniform(50, 90)
```

```
        pressure = temperature * np.random.uniform(1.8, 2.2) + np.random.normal(0, 3)
        water_level = (150 - temperature) * np.random.uniform(0.1, 0.15) + np.random.normal(0, 1)
        data.append([temperature, pressure, water_level])
    return np.array(data)
```

Метод створює випадкові значення параметрів температури, тиску та рівня води, які логічно пов’язані між собою. Це дозволяє симулювати нормальні умови роботи системи для тренування нейромережі. Фрагмент нижче відповідає за завантаження існуючої моделі або тренування нової з подальшим збереженням результатів.

```
def _load_or_train_model(self):
    if os.path.exists(MODEL_FILE) and os.path.exists(SCALER_FILE) and os.path.exists(THRESHOLD_FILE):
    else:
```

Метод перевіряє наявність раніше збережених файлів. Якщо вони є — виконується завантаження моделі, масштабувальника та порогу. Якщо ні — відбувається:

– нормалізація значень у діапазон [0, 1];

– розрахунок порогу аномалії як 98-го процентиля значень MSE.

Фрагмент коду нижче відповідає за логування подій, які відбуваються під час роботи програми. Він забезпечує запис текстових повідомлень у файл events.log, а також дозволяє зчитати історію подій із цього файлу для подальшого відображення у графічному інтерфейсі.

```
def log_event(message):
    with open("events.log", "a", encoding="utf-8") as f:
        f.write(f"{message}\n")
def load_events():
    try:
        with open("events.log", "r", encoding="utf-8") as f:
            lines = f.readlines()
        return lines
    except FileNotFoundError:
        return []
```

Функція log_event() додає новий рядок з повідомленням до файлу events.log. Вона відкриває файл у режимі дозапису (append) і додає повідомлення з нового рядка. Функція load_events() намагається зчитати всі рядки з файлу events.log. Якщо файл не існує, вона повертає порожній список — це запобігає аварійному

завершенню програми. У фрагменті нижче створюється графічний інтерфейс користувача для моніторингу системи.

```
class MonitoringApp:
    def __init__(self, root):
        self.root = root
        self.root.title
        self.root.geometry("600x600")
        self.root.resizable(False, False)
        self.default_font = font.nametofont("TkDefaultFont")
        self.default_font.configure(size=12)
        self.autoencoder = AutoencoderHandler()
        self.simulation_running = False
        self.attack_mode = False
```

Код містить інтерфейсні елементи для запуску симуляції, ввімкнення "кібератаки", відображення параметрів системи та виводу журналу подій.

Клас MonitoringApp створює вікно додатку з фіксованим розміром 600x600 пікселів і забороняє зміну його розмірів. У ньому ініціалізуються головні змінні, зокрема екземпляр класу AutoencoderHandler, логічні прапорці для статусу симуляції та режиму атаки. Також тут створюються кнопки запуску симуляції та атаки, а також текстові поля для відображення температури, тиску, рівня води, похибки та статусу. Код наведений нижче відображає роботу методу toggle_simulation() відповідає за запуск або зупинку симуляції за натиском кнопки.

```
def toggle_simulation(self):
        if not self.simulation_running:
            self.simulation_running = True
            self.start_button.config(text="Зупинити симуляцію")
            self.run_simulation()
        else:
            self.simulation_running = False
            self.start_button.config(text="Почати симуляцію")
```

Коли симуляція не активна, метод запускає її та змінює напис на кнопці на "Зупинити симуляцію". Якщо вона вже активна — зупиняє симуляцію та повертає початковий напис на кнопці. Код наведений нижче відображає роботу методу toggle_attack() перемикає стан між нормальним режимом і "кібератакою". Залежно від режиму змінюється напис на відповідній кнопці.

```
def toggle_attack(self):
```

```
        self.attack_mode = not self.attack_mode
        if self.attack_mode:
            self.attack_button.config(text="Відмінити атаку")
        else:
            self.attack_button.config(text="Кібератака")
```

Коли користувач активує атаку, напис змінюється на "Відмінити атаку", і навпаки. Це дає змогу симулювати ситуацію порушення безпеки. Код наведений нижче відображає роботу методу, який додає новий рядок до текстового журналу в інтерфейсі.

```
    def append_log(self, message, color="black"):
        self.log_area.config(state='normal')
self.log_area.tag_configure(color, foreground=color)
        self.log_area.insert("1.0", message + "\n", color)
        self.log_area.config(state='disabled')
```

Код наведений нижче відображає основну логіку симуляції, що знаходиться в методі run_simulation(). Вона виконується кожну секунду, створює нові вхідні дані, аналізує їх за допомогою автоенкодера, і відображає результати в інтерфейсі.

```
def run_simulation(self):
        if not self.simulation_running:
            return
```

Цей фрагмент перевіряє, чи активна симуляція. Якщо симуляція зупинена метод завершує виконання, запобігаючи непотрібним обчисленням. Код наведений нижче відображає генерацію даних залежно від вибраного режиму.

```
if self.attack_mode:
    stream_data = self.autoencoder.create_stream_data_array(1, 1)
    temperature = stream_data[0][0]
    pressure = stream_data[0][1]
    water_level = stream_data[0][2]
else:
    stream_data = self.autoencoder.create_stream_data_array(1, 0)
    temperature = stream_data[0][0]
    pressure = stream_data[0][1]
    water_level = stream_data[0][2]
```

Код вище генерує вхідні дані для аналізу. Цей блок генерує вхідні дані для аналізу. Залежно від активного режиму створюються різні типи даних:

–..при True генеруються аномальні дані (параметр 1 у другому аргументі);

–..при False генеруються нормальні дані (параметр 0 у другому аргументі).

Отримані значення температури, тиску та рівня води витягуються з першого рядка згенерованого масиву. Код наведений нижче відображає підготовку та аналіз даних.

```
input_data = [temperature, pressure, water_level]
result = self.autoencoder.analyze(input_data)
```

Код вище формує масив вхідних даних та передає їх автоенкодеру для аналізу.

Код наведений нижче відповідає за оновлення інтерфейсу користувача.

```
self.temp_var.set(f"{temperature:.2f} °C")
self.pressure_var.set(f"{pressure:.2f} kPa")
self.water_var.set(f"{water_level:.2f} %")
self.error_var.set(f"{result['mse']:.5f}")
```

Цей блок оновлює відображення поточних параметрів системи в графічному інтерфейсі. Значення температури, тиску, рівня води форматуються з відповідною точністю та одиницями вимірювання. Також відображаються розрахована помилка та статус системи. Код наведений нижче відповідає за обробку нормального стану.

```
if result['status'] == "АНОМАЛІЯ!":  self.status_label.config(foreground="red")
    log_event(f"[LOG][{datetime.now()}] АНОМАЛІЯ! Вхідні дані: {input_data}, Відновлені дані: {result['reconstructed']}, MSE: {result['mse']:.5f}")
    self.append_log(f"[LOG][{datetime.now()}] АНОМАЛІЯ! Вхідні дані: {input_data}, Відновлені дані: {result['reconstructed']}, MSE: {result['mse']:.5f}\n", color="red")
```

Блок коду вище обробляє випадок виявлення аномалії в системі. При виявленні аномального стану:

–..змінює колір статусної мітки на червоний для візуального попередження;

–..записує детальну інформацію про аномалію до системного логу;

–..додає запис до інтерфейсного логу з червоним кольором для привернення уваги оператора.

Код наведений нижче відповідає за планування наступного циклу.

```
self.root.after(1000, self.run_simulation)
```

Останній рядок коду забезпечує циклічне виконання симуляції. Через 1000 мілісекунд метод run_simulation буде викликаний знову, створюючи

безперервний цикл моніторингу системи. Це дозволяє отримувати актуальні дані в реальному часі.

3.3 Тестування ПЗ виявлення атак на промислові системи керування ICS з використанням автоенкодерів та інструкція для користувача

З метою перевірки працездатності реалізованої системи виявлення атак на промислові системи керування проведено експериментальне тестування з використанням моделі автоенкодера, навченої на нормальних даних. Система призначена для моніторингу технологічних параметрів, зокрема температури, тиску та рівня води в умовному резервуарі. Для кожного нового зчитаного набору параметрів здійснюється обчислення середньоквадратичної похибки (MSE) між фактичними та відновленими значеннями.

Розроблене програмне забезпечення для виявлення атак на промислові системи керування (ICS) реалізує механізм машинного навчання на основі автоенкодера, що працює з трьома ключовими технологічними параметрами: тиском, температурою та рівнем води. Під час попереднього навчання автоенкодер опрацьовував лише «нормальні» (неаномальні) дані. Після завершення навчання система переходить у режим реального часу, в якому вона аналізує нові вхідні дані. Програма обчислює відновлені значення параметрів за допомогою декодера та порівнює їх із фактичними показниками. Основним індикатором аномалії виступає похибка відновлення – середньоквадратичне відхилення (Mean Squared Error). Якщо значення похибки перевищує заздалегідь визначений поріг, програма інтерпретує таку поведінку як потенційну кібератаку або технічну аномалію, виводячи відповідне повідомлення.

Початковий інтерфейс програмного забезпечення відображає порожні поля для приймання показників тиску, температури та рівня води. Також присутній блок виведення похибки, яка використовується для оцінки стану системи, та текстове поле зі статусом, що демонструє поточний режим роботи (норма або

атака). Інтерфейс є зручним для користувача і забезпечує візуальний контроль за параметрами у реальному часі, ще до початку активного процесу моніторингу.

Початкове вікно програмного забезпечення містить зручний інтерфейс для користувача, який дає змогу візуально оцінювати поточний стан системи та керувати симуляцією даних. На екрані відображаються такі основні елементи:

– три поля введення/відображення параметрів: температура, тиск і рівень води – зчитуються системою автоматично та оновлюються в режимі реального часу;

– поле «Похибка» (MSE) – показує обчислену середньоквадратичну похибку між вхідними та відновленими значеннями. Цей індикатор допомагає оцінити, чи відбувається нормальна робота системи;

– поле «Статус» – виводить поточний стан системи: «норма» або «атака», залежно від величини похибки;

– журнал подій – розташований нижче основних полів; відображає хронологію подій, включно з моментами зміни режиму та виявлення аномалій.

Після запуску програмного забезпечення користувач потрапляє на головне вікно інтерфейсу (рис.3.1).

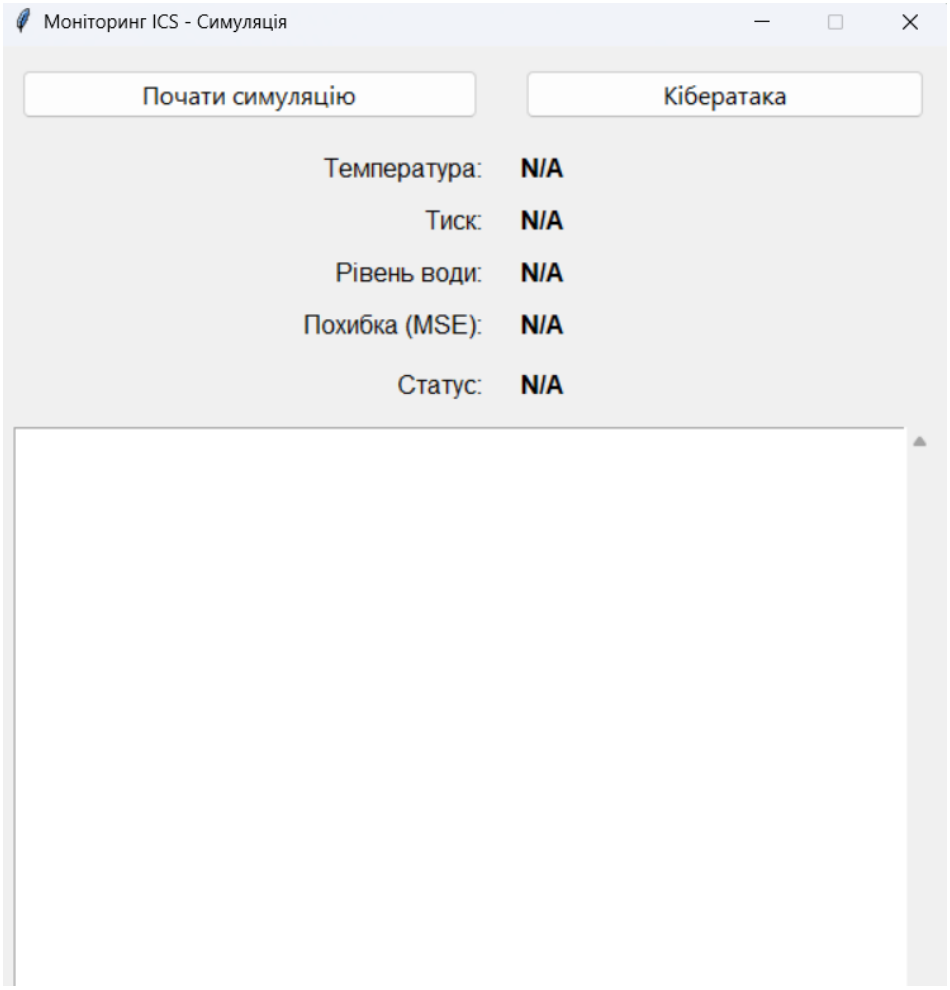


Рисунок 3.1 – Початкове вікно програмного засобу для виявлення атак на ICS з використанням автоенкодера

Також у початковому вікні розміщено дві основні кнопки керування розробленою програмою виявлення атак на ICS:

– кнопка «Почати симуляцію» запускає штатний режим моделювання нормальної роботи системи, якщо натиснути цю кнопку то ініціюється тестова подача нормальних (неаномальних) даних, так система буде обчислювати похибку та оновлювати статус – цей режим корисний для перевірки, чи правильно працює автоенкодер;

– кнопка «Кібератака» - активує сценарій з аномальними параметрами, що імітує кібератаку або збій у технологічному процесі, а якщо користувач натиснить цю кнопку, то перевірить реакцію системи на підміну значень у системі, тобто якщо система працює коректно, вона повинна виявити аномалію,

змінити статус на «аномалія», збільшити значення похибки та зафіксувати інцидент у журналі подій.

У наступному етапі, демонструється робота програми в штатному (пасивному) режимі. Система приймає вхідні показники, здійснює їх обробку за допомогою автоенкодера, і, виявивши, що похибка залишається в межах допустимої норми, змінює статус на «норма». Це означає, що всі дані відповідають раніше вивченим шаблонам і в системі не фіксується підозрілих відхилень.

Крім того, програмне забезпечення автоматично записує кожен цикл обробки до журналу подій, який знаходиться під блоком виводу основних параметрів. Така функціональність дозволяє не лише виявляти аномалії в режимі реального часу, а й здійснювати подальший аналіз інцидентів.

Така функціональність дозволяє не лише виявляти аномалії в режимі реального часу, а й здійснювати подальший аналіз інцидентів. Система оновлює показники щосекунди, забезпечуючи безперервний моніторинг стану промислового обладнання та своєчасне реагування на будь-які відхилення від норми. Робота програмного засобу в режимі реального часу без виявлення аномалій (рис.3.2).

На представленому зображені видно поточні параметри системи: температура становить 53.05 °C, тиск - 104.12 kPa, а рівень води - 13.11 %. Значення похибки MSE дорівнює 0.11388, що свідчить про невелике відхилення від еталонних показників, проте воно залишається в межах допустимого діапазону. Статус системи відображається зеленим кольором, що візуально підтверджує нормальний стан роботи.

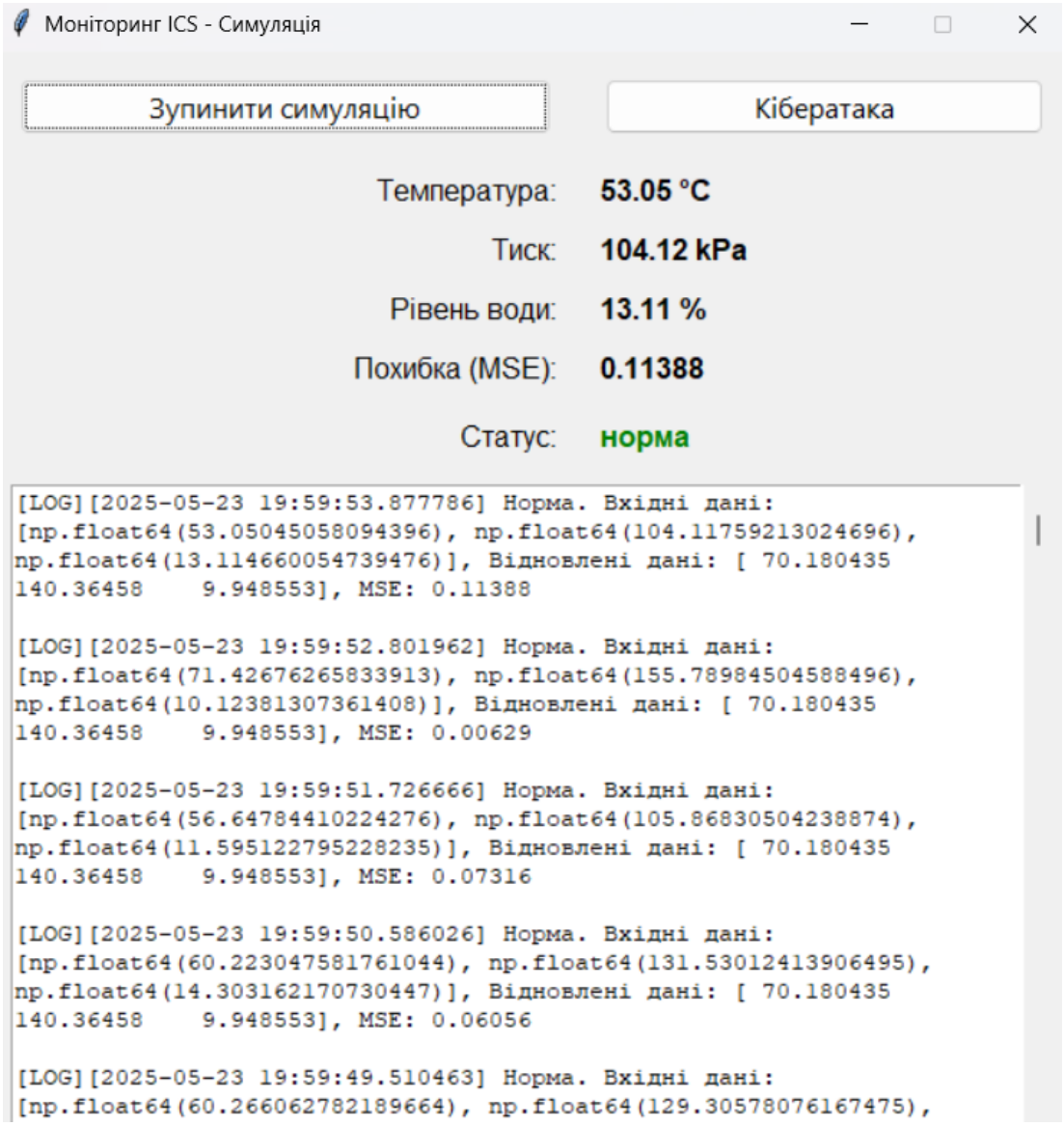


Рисунок 3.2 – Вікно роботи програмного засобу в режимі реального часу без виявлення аномалій

На третьому етапі експерименту здійснено імітацію атаки з підміною значень технологічних параметрів. Натискання кнопки «Кібератака» активує режим атаки: у систему подаються навмисно змінені (аномальні) значення параметрів. У результаті цього збільшується похибка, що має призвести до автоматичного визначення стану «атака». Така симуляція дає змогу протестувати ефективність детектора та його здатність виявляти підміну даних.

Після надходження аномальних даних система фіксує значне зростання похибки реконструкції, яка перевищує поріг нормального функціонування. Алгоритм негайно реагує, змінюючи статус на «атака» та виділяючи проблему в

інтерфейсі. Це зумовлено принципом роботи автоенкодера: він не здатен ефективно відновити дані, які не відповідають нормальним закономірностям (наприклад, коли висока температура поєднується з низьким тиском і підвищеним рівнем води). Таким чином, система демонструє здатність до виявлення навіть складних багатофакторних аномалій, типових для ICS-середовищ. Крім того, інформація про зміну режиму одразу записується у журнал подій, що дозволяє відстежувати історію змін. Вікно вигляду виявлення атаки з підміною значень системою на основі автоенкодера (рис.3.3).

На представленому зображені зафіксовано момент активації режиму кібератаки. Параметри системи демонструють явні відхилення від нормальних значень: температура досягла критичного рівня 105.05 °C, тиск зріс до 229.83 kPa, а рівень води підскочив до 97.63 %. Такі показники є фізично неможливими в нормальних умовах експлуатації промислового обладнання та свідчать про спробу маніпуляції даними.

Після надходження аномальних даних система фіксує значне зростання похибки реконструкції до 15.83378, яка значно перевищує поріг нормального функціонування (порівняно з попереднім значенням 0.11388). Алгоритм негайно реагує, змінюючи статус на «АНОМАЛІЯ!» та виділяючи проблему червоним кольором в інтерфейсі. Це зумовлено принципом роботи автоенкодера: він не здатен ефективно відновити дані, які не відповідають нормальним закономірностям. У журналі подій чітко відображено зміну характеру записів - всі нові повідомлення позначені як "АНОМАЛІЯ!" та містять детальну інформацію про підроблені вхідні дані, відновлені значення та розрахунки похибки.

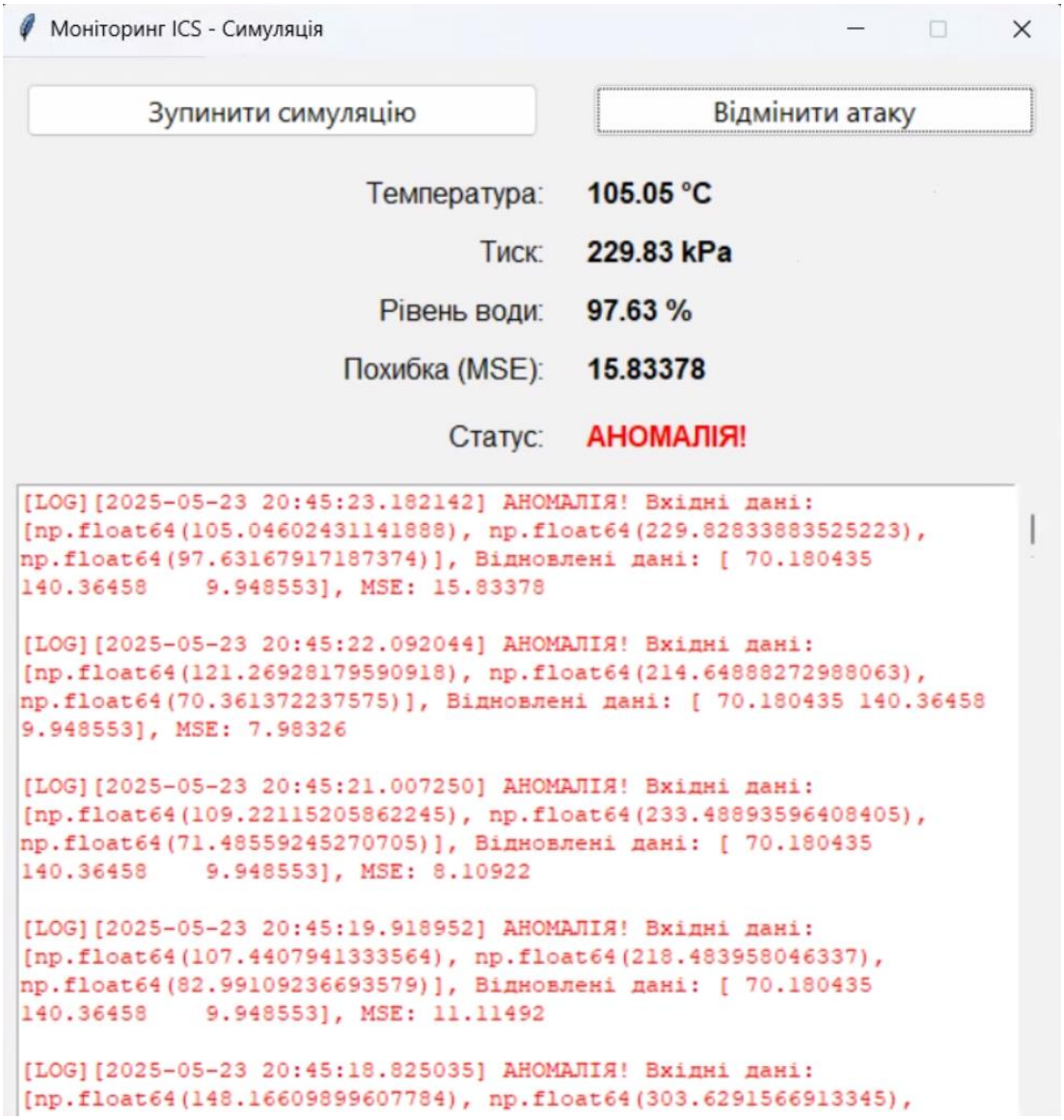


Рисунок 3.3 – Вигляд вікна виявлення атаки з підміною значень системою на основі автоенкодера

Такий підхід дозволяє системі демонструвати здатність до виявлення навіть складних багатофакторних аномалій, типових для ICS-середовищ. Після завершення симуляції можна повторно натиснути «Зупинити симуляцію», щоб деактивувати режим атаки, або перемкнутися між режимами для порівняння результатів. Журнал подій зберігає повну історію всіх операцій, дозволяючи здійснити ретроспективний аналіз та документування інцидентів безпеки.

Фінальний етап експерименту показує повернення до початкового стану після завершення атаки. На екрані знову відображаються поля з параметрами, як у першому запуску, але тепер також міститься інформація з попередніх

сеансів тестування. Це дозволяє користувачу проаналізувати історію подій та переконатися у відновленні нормальної роботи системи. Такий підхід забезпечує прозорість функціонування програмного забезпечення та дає змогу проводити ретроспективну оцінку стану системи за різні проміжки часу. Початкове вікно перед повторним запуском, із збереженням даних попередніх сеансів (рис. 3.4).

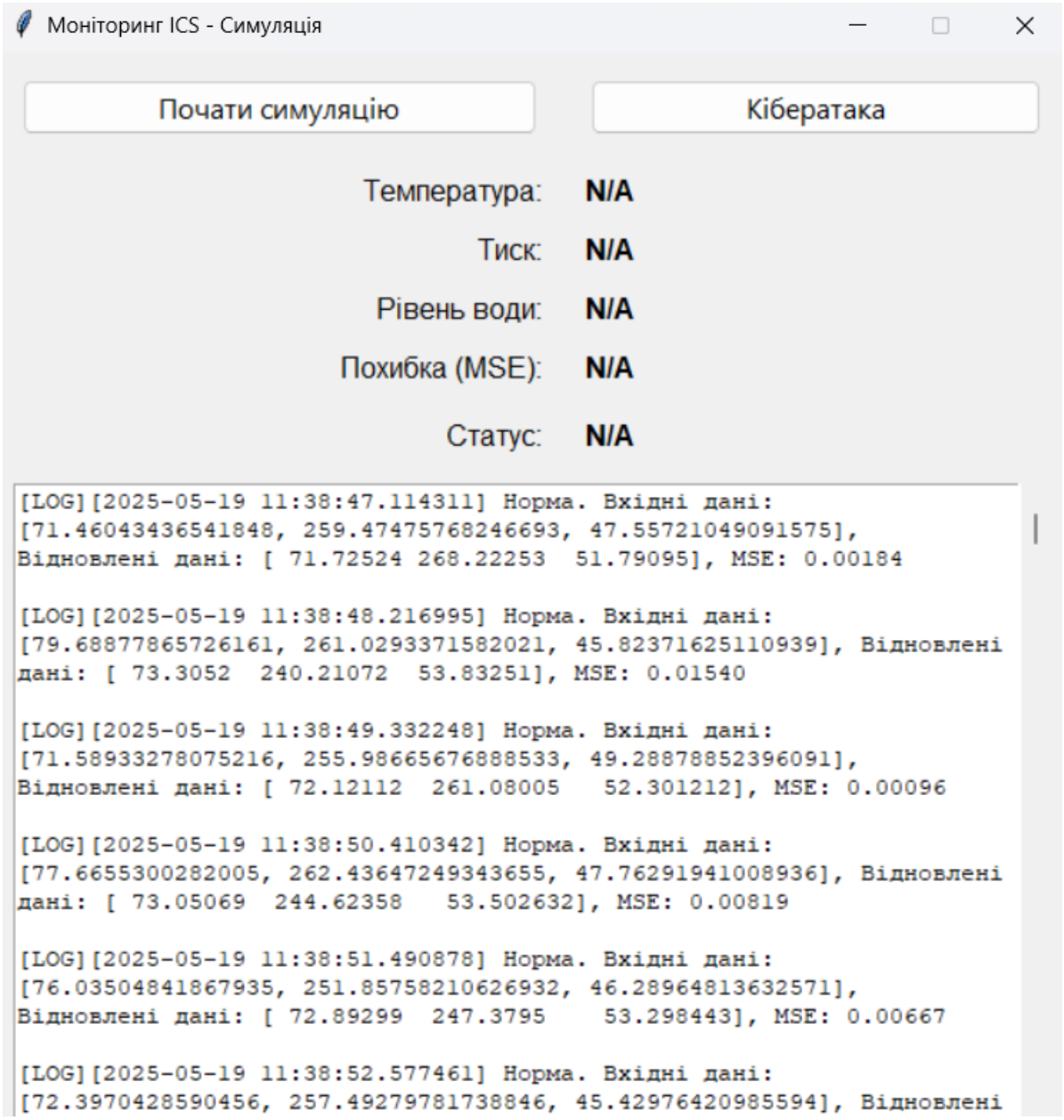


Рисунок 3.4 – Початкове вікно перед повторним запуском, із збереженням даних попередніх сеансів

Проведено тестування програмного забезпечення для виявлення атак на промислові системи керування (ICS), що використовує автоенкодери для виявлення аномалій у технологічних параметрах. Реалізована система забезпечує моніторинг таких показників, як температура, тиск і рівень води в умовному резервуарі, з подальшим розрахунком середньоквадратичної похибки (MSE) між вхідними та відновленими значеннями. Отримана похибка є ключовим критерієм для визначення наявності або відсутності аномального функціонування. Таким чином, у разі перевищення допустимого порогу похибки, система формує відповідне повідомлення про потенційну атаку у журнал подій.

У результаті проведеного експерименту встановлено, що розроблена система на основі автоенкодера ефективно виконує виявлення аномалій у даних промислового процесу. Вона демонструє чутливість до змін у параметрах, оперативно реагує на відхилення та переходить до стабільного режиму після відновлення нормальної роботи. Запропонований підхід може бути використаний як складова частина системи кіберзахисту для раннього виявлення інцидентів безпеки в середовищах критичної інфраструктури.

3.5 Висновки до розділу 3

У цьому розділі було проведено програмну реалізацію системи виявлення атак на промислові системи керування (ICS) з використанням автоенкодерів. Розроблений програмний засіб враховує специфіку промислових мереж та вимоги до своєчасного виявлення кібератак на критичну інфраструктуру.

На основі порівняльного аналізу мов програмування та фреймворків було обґрунтовано вибір технологічного стеку Python, TensorFlow та Tkinter, який забезпечує оптимальне поєднання продуктивності, простоти розробки та наявності потужних бібліотек машинного навчання. Архітектура програмного засобу побудована на модульному принципі з чітким розділенням

функціональності між компонентами обробки даних, машинного навчання та користувацького інтерфейсу.

Тестування ефективності розробленої системи показало її здатність до успішного виявлення аномалій у технологічних параметрах промислових систем. Система демонструє стабільну роботу в режимі реального часу з автоматичним логуванням подій та візуальною індикацією стану безпеки. Експериментальна перевірка підтвердила, що автоенкодер ефективно розпізнає як поодинокі викиди параметрів, так і складні багатофакторні аномалії, характерні для цілеспрямованих кібератак на ICS-середовища.

У результаті розробки програмного засобу було створено функціональний прототип модуля систем керування на ICS-середовищі, який може служити основою для подальшого розвитку та впровадження в реальних промислових умовах. Розроблена система забезпечує безперервний моніторинг стану обладнання та автоматичне виявлення потенційних загроз кібербезпеці критичної інфраструктури.

ВИСНОВОК

У першому розділі було розглянуто теоретичні основи виявлення атак на промислові системи керування (ICS). Проаналізовано базові поняття кібербезпеки в контексті ICS, класифікацію типових атак на такі системи, а також існуючі методи виявлення аномалій, зокрема на основі сигнатурного та поведінкового (аномалійного) аналізу. Досліджено досвід застосування автоенкодерів та інших архітектур нейронних мереж для виявлення аномалій у промислових середовищах, а також виконано огляд існуючих рішень та програмних засобів, що використовуються в галузі. На основі проведеного аналізу було сформульовано завдання дослідження, яке передбачало створення ефективного підходу до виявлення аномалій на ICS за допомогою автоенкодерів.

У другому розділі було запропоновано підхід до виявлення аномалій у промислових системах керування з використанням автоенкодеру. Обґрунтовано доцільність вибору саме цієї архітектури нейромережі, оскільки вона дозволяє виявляти відхилення у поведінці системи без потреби в детальній попередній класифікації атак. Спроєктовано архітектуру модуля обробки навчального датасету, алгоритм навчання автоенкодеру, а також загальну архітектуру програмного засобу, який реалізує запропонований підхід. Під час проектування враховувалися вимоги до ефективності, адаптивності та простоти інтеграції в існуючі ICS-інфраструктури.

У третьому розділі було реалізовано програмний засіб для виявлення атак на ICS з використанням автоенкодерів. Обґрунтовано вибір мови програмування Python та бібліотек TensorFlow і Scikit-learn, які забезпечують ефективну реалізацію і навчання нейронної мережі. Описано структуру функціональних модулів ПЗ, проведено тестування точності виявлення аномалій, а також складено інструкцію для користувачів, яка пояснює принципи роботи з програмою. Результати експериментів підтвердили ефективність розробленого підходу у виявленні аномальних станів системи на основі аналізу

історичних даних ICS.

У результаті виконання бакалаврської кваліфікаційної роботи було успішно реалізовано програмний засіб для виявлення атак на промислові системи керування з використанням автоенкодеру. Запропонований підхід дозволяє виявляти невідомі раніше загрози та відхилення від нормальної поведінки системи, що сприяє підвищенню рівня кіберзахисту ICS.

Спираючись на результати проведених робіт, реалізовану програмну архітектуру та експериментальну перевірку, можна стверджувати, що в межах даної бакалаврської кваліфікаційної роботи було досягнуто поставлену мету та виконано всі задачі дослідження.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lindsay J. R. Stuxnet revisited: From cyber warfare to secret statecraft. Journal of Strategic Studies. 2025. P. 1–40.
2. Смольянінов П. О., Кравцов Г. О. Огляд індустріальних систем управління. Моделювання та інформаційні технології. 2017. № 80. С. 70–80.
3. Зубок В. Ю.; Гончар, С. Ф.; Єрмошин, В. В.; Карасюк, Г. О. та ін. Архітектурно-функціональне порівняння відомих платформ та систем кіберзахисту промислових об’єктів. Èlektronnoe Modelirovanie. 2022. Т. 44. № 3. p. 65.
4. Zahran B., Abu Zahra F. IT/OT Convergence Protocols: DNP3, Ethernet/IP, and Modbus. 2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE). IEEE, 2023. p. 1743-1748.
5. Orlova, Nataliia; Mokhova, Iuliia. Впровадження інформаційних технологій в систему корпоративного управління. Електронне наукове фахове видання “ВІДКРИТЕ ОСВІТНЄ Е-СЕРЕДОВИЩЕ СУЧАСНОГО УНІВЕРСИТЕТУ”, 2017, 3: С. 355-365.
6. Ольшевський С. І., та ін. Сучасні інтерфейси людина-машина у складі відокремлених систем керування. 2023. С. 335.
7. Єсіна, О. Г., Лінгур Л. М. Проблеми впровадження та використання інформаційних технологій на підприємстві / О. Г. Єсіна, Л. М. Лінгур // Науковий вісник Ужгородського національного університету : серія: Міжнародні економічні відносини та світове господарство / голов. ред. М.М. Палінчак. – Ужгород : Гельветика, 2019. – Вип. 24, №Ч.2. – С. 16–20. – Рез. рос., англ. – Бібліогр. : с. 19–20
8. Атанасов, М.; Йона, О. Вплив інформаційних технологій на розвиток підприємства. In: Гармонізація суспільства–новітній напрямок розвитку держави: Всеукр. наук. конф. аспірантів та молодих вчених. 2014. с. 54-61.
9. Грудзинський Ю. Є., Шулепа А. М. Особливості оцінки ризику в автоматизованих системах керування технологічними процесами. Вісник

Національного технічного університету ХПІ. Серія: Нові рішення в сучасних технологіях. 2018. № 16. С. 117–121.

10. Єрмошин В. В., Карасюк Г. О., Гончар С. Ф. Дослідження кіберризиків автоматизованих систем управління технологічними процесами. Èlektronnoe Modelirovanie. 2022. Т. 44. № 1. С. 14.

11. Суходоля О. М. Захист критичної інфраструктури в умовах гібридної війни: проблеми та пріоритети державної політики України. Стратегічні пріоритети. 2016. № 3. С. 62–76.

12. Смольянінов П. О., Кравцов Г. О. Огляд індустріальних систем управління. Моделювання та інформаційні технології. 2017. № 80. С. 70–80.

13. Yakoviv I., Tsyganok V. V. A Procedure for Assessing the State of Cybersecurity of Power Grids. ITS. 2018.

14. Tkachuk, Nikolay; Sokol, Vladyslav; Glukhovtsova, Kateryna. An Integrated Development Framework for Advanced IT-Service Management: Proof-of-Concept Project in Universities Domain. In: Information and Communication Technologies in Education, Research, and Industrial Applications: 9th International Conference, ICTERI 2013, Kherson, Ukraine, June 19-22, 2013, Revised Selected Papers 9. Springer International Publishing, 2013. p. 50-69.

15. Захаров, В. А. Операційна система промислового підприємства. Бізнес Інформ, 2015, 11: 400-405.

16. Волошин, А. Л.; Криховецький, Г. Я. Метод модернізації комплексу засобів захисту інформації від несанкціонованого доступу в сучасних автоматизованих системах. Системи обробки інформації, 2014, 1: 142-147.

17. Eslahi, Meisam, et al. BYOD: Current state and security challenges. In: 2014 IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE). IEEE, 2014. p. 189-192.

18. Яковів І., Циганок В. Аналіз процедури оцінювання стану кібервразливості систем електропостачання. Інформаційні технології та безпека. 2018. С. 286.

19. Igure, Vinay M.; Laughter, Sean A.; Williams, Ronald D. Security issues in SCADA networks. computers & security, 2006, 25.7: 498-506.

20. Douligeris C., Mitrokotsa A. DDoS attacks and defense mechanisms: classification and state-of-the-art. Computer Networks. 2004. Vol. 44, No. 5. P. 643–666.

21. Yadav G., Paul K. Architecture and security of SCADA systems: A review. International Journal of Critical Infrastructure Protection. 2021. Vol. 34. P. 100433.

22. Полікаровських О. І., Малаксіано М. О., Даус Ю. В. Протидія кібернетичним атакам на морському транспорті. Вісник Одеського національного морського університету. 2024. No 73. С. 234–247.

23. Yakoviv I., Trokhymenko A., Hlum K. Спосіб визначення каналу керування APT-атакою. Information Technology and Security. 2021. Vol. 9, No. 2. P. 176–188.

24. Lakhno V., et al. WAF захисту внутрішніх сервісів у структурі Zero Trust. Кібербезпека: освіта, наука, техніка. 2021. No 1.13. С. 81–91.

25. Мєшков В. Аналіз систем інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак. Information Technology: Computer Science, Software Engineering and Cyber Security. 2023. No 1. С. 85–92.

26. Chakravarty A. K., et al. A study of signature-based and behaviour-based malware detection approaches. Int. J. Adv. Res. Ideas Innov. Technol. 2019. P. 1509–1511.

27. Behal S., Brar A. S., Kumar K. Signature-based botnet detection and prevention. Proceedings of International Symposium on Computer Engineering and Technology. 2010. P. 127–132.

28. Cortes C., Pregibon D. Signature-based methods for data streams. Data Mining and Knowledge Discovery. 2001. Vol. 5, No. 3. P. 167–182.

29. Hubballi N., Suryanarayanan V. False alarm minimization techniques in signature-based intrusion detection systems: A survey. Computer Communications. 2014. Vol. 49. P. 1–17.

30. Bangui H., Ge M., Buhnova B. A hybrid machine learning model for intrusion detection in VANET. Computing. 2022. P. 503–531.

31. Yang Z., et al. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. Computers & Security. 2022. Vol. 116. P. 102675.

32. Garcia-Teodoro P., et al. Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security. 2009. Vol. 28, No. 1–2. P. 18–28.

33. Garcia-Teodoro P., et al. Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security. 2009. P. 18–28.

34. Wang K., Stolfo S. J. Anomalous payload-based network intrusion detection. International Workshop on Recent Advances in Intrusion Detection. Berlin, Heidelberg : Springer, 2004. P. 203–222.

35. Jyothsna V., Prasad K. M. Anomaly-Based Intrusion. Computer and Network Security. 2020. P. 35-50.

36. Davis J. J., Clark A. J. Data preprocessing for anomaly based network intrusion detection: A review. Computers & Security. 2011. Vol. 30, No. 6–7. P. 353–375.

37. Hu Y., et al. Network data analysis and anomaly detection using CNN technique for industrial control systems security. 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). IEEE, 2019. p. 593-597.

38. Rick R., Berton L. Energy forecasting model based on CNN-LSTM-AE for many time series with unequal lengths. Engineering Applications of Artificial Intelligence. 2022. Vol. 113. P. 104998.

39. Hnamte V., et al. A novel two-stage deep learning model for network intrusion detection: LSTM-AE. IEEE Access. 2023. Vol. 11. P. 37131–37148.

40. Shiri F. M., et al. A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU. arXiv preprint arXiv:2305.17473. 2023. P. 67.

41. Park W., Ahn S. Performance comparison and detection analysis in Snort and Suricata environment. Wireless Personal Communications. 2017. Vol. 94. P. 241–252.

42. Boukebous A. A. E., et al. A comparative analysis of Snort 3 and Suricata. 2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET). IEEE, 2023. P. 1-6.

43. Totaro S. Integrazione di Nozomi Networks e Splunk per la raccolta e l'analisi di dati provenienti da dispositivi di tipo OT e IoT = Nozomi Networks and Splunk integration for data collection and analysis from OT and IoT devices : diss. Politecnico di Torino, 2023. P. 101.

44. Team82, Claroty. Claroty Biannual ICS Risk & Vulnerability Report: 1H 2021. Claroty, UK, 2021. P. 42.

45. Barda I. Just about managing: dealing with the threats to industrial systems. Computer Fraud & Security. 2019. No. 9. P. 11–14.

46. Waleed A., Jamali A. F., Masood A. Which open-source IDS? Snort, Suricata or Zeek. Computer Networks. 2022. Vol. 213. P. 109116.

47. Hurd C. M., McCarty M. V. A survey of security tools for the industrial control system environment. Idaho National Laboratory (INL). Report No. INL/EXT--17-42229. 2017. P. 42.

48. Ghahramani Z. Unsupervised learning. Summer School on Machine Learning. Berlin, Heidelberg : Springer, 2003. P. 72–112.

ДОДАТКИ

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
Юрій ЯРЕМЧУК
" 20 " березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської кваліфікаційна роботи на тему:

зробка програмного засобу для виявлення атак на промислові системи керування (ICS) з використанням автоенкодерів

08–72.БДР.019.00.092.ТЗ

Керівник бакалаврської кваліфікаційної роботи
д.ф. (PhD)доц., каф. МБІС
Салієва О. В.
" "

Вінниця – 2025 р.

1. Найменування та область застосування

Розробка програмного засобу для виявлення атак на промислові системи керування (ICS) з використанням автоенкодерів

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 90 від 20.03.2025 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробити програмний засіб для виявлення атак на промислових системах керування (ICS)с з використанням автоенкодерів

3.2 Призначення: розроблений програмний засіб забезпечує виявлення атак на промислові системи керування (ICS)с з використанням автоенкодерів

4. Джерела розробки

4.1. Єсіна, О. Г., Лінгур Л. М. Проблеми впровадження та використання інформаційних технологій на підприємстві / О. Г. Єсіна, Л. М. Лінгур // Науковий вісник Ужгородського національного університету : серія: Міжнародні економічні відносини та світове господарство / голов. ред. М.М. Палінчак. – Ужгород : Гельветика, 2019. – Вип. 24, №Ч.2. – С. 16–20.

4.2. Смольянінов П. О., Кравцов Г. О. Огляд індустріальних систем управління. Моделювання та інформаційні технології. 2017. № 80. С. 70–80.

4.3. Суходоля О. М. Захист критичної інфраструктури в умовах гібридної війни: проблеми та пріоритети державної політики України. Стратегічні пріоритети. 2016. № 3. С. 62–76.

4.4. Грудзинський Ю. Є., Шулепа А. М. Особливості оцінки ризику в автоматизованих системах керування технологічними процесами. Вісник Національного технічного університету ХПІ. Серія: Нові рішення в сучасних технологіях. 2018. № 16. С. 117–121.

4.5. Яковів І., Циганок В. Аналіз процедури оцінювання стану кібервразливості систем електропостачання. Інформаційні технології та безпека. 2018. С. 286.

4.6. Мєшков В. Аналіз систем інтелектуального моніторингу трафіку

комп’ютерної мережі для систем виявлення атак. Information Technology: Computer Science, Software Engineering and Cyber Security. 2023. No 1. С. 85–92.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Intel Core i3–3800U 1.80GHz і подібні до них;

– оперативна пам’ять – не менше 2Gb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.3

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко–економічні показники

8.1 Цінність результатів використання даного проекту повинна евищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для ористання широкого загалу.

9. Стадії та етапи розробки

п	Назва етапів бакалаврської кваліфікаційної роботи	Початок	Закінчення
	Визначення напрямку бакалаврської роботи, формулювання теми	21.03.2025	28.03.2025
	Аналіз предметної області обраної теми	28.03.2025	14.04.2025
	Розробка алгоритму роботи	15.04.2025	22.04.2025
	Написання бакалаврської роботи на основі розробленої теми	23.04.2025	16.05.2025
	Передзахист бакалаврської кваліфікаційної роботи	16.05.2025	26.05.2025
	Виправлення, уточнення, корегування бакалаврської кваліфікаційної роботи	27.05.2025	04.06.2025
	Захист бакалаврської кваліфікаційної роботи	09.06.2025	11.06.2025

10. Порядок контролю та прийому

10.1 До приймання бакалаврської кваліфікаційної роботи надається:

– ПЗ до бакалаврської кваліфікаційної роботи;

- демонстрація результату бакалаврської кваліфікаційної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента

Технічне завдання до виконання прийняв _____ Гідлуцький О. А.

Додаток Б. Лістинг коду програмного засобу

```
import tkinter as tk
from tkinter import ttk, font, scrolledtext
from model_handler import AutoencoderHandler
import numpy as np
from logger import log_event, load_events
from datetime import datetime
import numpy as np
import os
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.callbacks import EarlyStopping

MODEL_FILE = "autoencoder_model.keras"
SCALER_FILE = "scaler.npy"
THRESHOLD_FILE = "threshold.npy"

class AutoencoderHandler:
    def __init__(self):
        self.model = None
        self.scaler = MinMaxScaler()
        self.threshold = None
        self._load_or_train_model()

    def _create_training_data_array(self, length):
        data = []
        for _ in range(length):
            temperature = np.random.uniform(50, 90)
            # Тиск зростає з температурою
            pressure = temperature * np.random.uniform(1.8, 2.2) + np.random.normal(0, 3)
            # Рівень води зменшується з температурою
            water_level = (150 - temperature) * np.random.uniform(0.1, 0.15) + np.random.normal(0, 1)
            data.append([temperature, pressure, water_level])
        return np.array(data)

    def _load_or_train_model(self):
        if os.path.exists(MODEL_FILE) and os.path.exists(SCALER_FILE) and os.path.exists(THRESHOLD_FILE):
            print("Завантаження моделі та параметрів...")
            self.model = load_model(MODEL_FILE)
            scaler_params = np.load(SCALER_FILE, allow_pickle=True).item()
            self.scaler.min_ = scaler_params['min']
            self.scaler.scale_ = scaler_params['scale']
            self.scaler.data_min_ = scaler_params['data_min']
            self.scaler.data_max_ = scaler_params['data_max']
            self.scaler.data_range_ = scaler_params['data_range']
            self.threshold = np.load(THRESHOLD_FILE)
        else:
            print("Навчання нової моделі...")
            training_data = self._create_training_data_array(5000)
            data_scaled = self.scaler.fit_transform(training_data)

            input_dim = data_scaled.shape[1]
            input_layer = Input(shape=(input_dim,))
            encoded = Dense(8, activation='relu')(input_layer)
            encoded = Dense(4, activation='relu')(encoded)
```

```
        bottleneck = Dense(2, activation='relu')(encoded)
        decoded = Dense(4, activation='relu')(bottleneck)
        decoded = Dense(8, activation='relu')(decoded)
        output_layer = Dense(input_dim, activation='linear')(decoded)

        self.model = Model(inputs=input_layer, outputs=output_layer)
        self.model.compile(optimizer='adam', loss='mse')
        early_stop = EarlyStopping(monitor='loss', patience=20, restore_best_weights=True)
        self.model.fit(data_scaled, data_scaled, epochs=500, verbose=0, callbacks=[early_stop])

        reconstructions_train = self.model.predict(data_scaled)
        mse_train = np.mean(np.power(data_scaled - reconstructions_train, 2), axis=1)
        self.threshold = np.percentile(mse_train, 98)

        self.model.save(MODEL_FILE)
        scaler_params = {
            'min': self.scaler.min_,
            'scale': self.scaler.scale_,
            'data_min': self.scaler.data_min_,
            'data_max': self.scaler.data_max_,
            'data_range': self.scaler.data_range_
        }
        np.save(SCALER_FILE, scaler_params)
        np.save(THRESHOLD_FILE, self.threshold)

    def analyze(self, input_data):
        scaled = self.scaler.transform([input_data])
        recon = self.model.predict(scaled)
        mse = np.mean(np.power(scaled - recon, 2))
        status = "АНОМАЛІЯ!" if mse > self.threshold else "норма"
        percent_error = round(mse * 100, 2)
        return {
            'input': input_data,
            'reconstructed': self.scaler.inverse_transform(recon)[0],
            'mse': mse,
            'percent_error': percent_error,
            'status': status
        }

    def create_stream_data_array(self, length, anomaly_p):
        data = []
        for _ in range(length):
            if np.random.rand() < anomaly_p:
                # Атака: дуже висока температура + високий рівень води
                temperature = np.random.uniform(100, 150)
                pressure = temperature * np.random.uniform(1.8, 2.2) + np.random.normal(0, 5)
                water_level = np.random.uniform(70, 100)  # занадто високий, автоматична подача
            else:
                temperature = np.random.uniform(50, 90)
                pressure = temperature * np.random.uniform(1.8, 2.2) + np.random.normal(0, 3)
                water_level = (150 - temperature) * np.random.uniform(0.1, 0.15) + np.random.normal(0, 1)
            data.append([temperature, pressure, water_level])
        return np.array(data)

class MonitoringApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Моніторинг ICS - Симуляція")
```

```
        self.root.geometry("600x600")  # Задаємо розмір вікна
        # fix the window size
        self.root.resizable(False, False)  # Забороняємо змінювати розмір вікна

        # Збільшуємо шрифти
        self.default_font = font.nametofont("TkDefaultFont")
        self.default_font.configure(size=12)

        self.autoencoder = AutoencoderHandler()
        self.simulation_running = False
        self.attack_mode = False

        self.start_button = ttk.Button(root, text="Почати симуляцію", command=self.toggle_simulation)
        self.start_button.grid(row=0, column=0, padx=15, pady=15, sticky="ew")

        self.attack_button = ttk.Button(root, text="Кібератака", command=self.toggle_attack)
        self.attack_button.grid(row=0, column=1, padx=15, pady=15, sticky="ew")

        label_opts = {"font": ("Arial", 12)}
        value_opts = {"font": ("Arial", 12, "bold")}

        ttk.Label(root, text="Температура:", **label_opts).grid(row=1, column=0, sticky="e", padx=10, pady=5)
        self.temp_var = tk.StringVar(value="N/A")
        ttk.Label(root, textvariable=self.temp_var, **value_opts).grid(row=1, column=1, sticky="w", padx=10,
pady=5)

        ttk.Label(root, text="Тиск:", **label_opts).grid(row=2, column=0, sticky="e", padx=10, pady=5)
        self.pressure_var = tk.StringVar(value="N/A")
        ttk.Label(root, textvariable=self.pressure_var, **value_opts).grid(row=2, column=1, sticky="w", padx=10,
pady=5)

        ttk.Label(root, text="Рівень води:", **label_opts).grid(row=3, column=0, sticky="e", padx=10, pady=5)
        self.water_var = tk.StringVar(value="N/A")
        ttk.Label(root, textvariable=self.water_var, **value_opts).grid(row=3, column=1, sticky="w", padx=10,
pady=5)

        ttk.Label(root, text="Похибка (MSE):", **label_opts).grid(row=4, column=0, sticky="e", padx=10, pady=5)
        self.error_var = tk.StringVar(value="N/A")
        ttk.Label(root, textvariable=self.error_var, **value_opts).grid(row=4, column=1, sticky="w", padx=10,
pady=5)

        ttk.Label(root, text="Статус:", **label_opts).grid(row=5, column=0, sticky="e", padx=10, pady=5)
        self.status_var = tk.StringVar(value="N/A")
        self.status_label = ttk.Label(root, textvariable=self.status_var, **value_opts)
        self.status_label.grid(row=5, column=1, sticky="w", padx=10, pady=10)

        root.grid_columnconfigure(0, weight=1)
        root.grid_columnconfigure(1, weight=1)

        self.log_area = scrolledtext.ScrolledText(root, wrap=tk.WORD, state='disabled')
        self.log_area.grid(row=6, column=0, columnspan=2, padx=10, pady=5, sticky="nsew")

        events = list(reversed(load_events()))
        for event in events:
            if "АНОМАЛІЯ!" in event:
                self.append_log(event.strip() + "\n", color="red")
            else:
                self.append_log(event.strip() + "\n")
```

```
    def toggle_simulation(self):
        if not self.simulation_running:
            self.simulation_running = True
            self.start_button.config(text="Зупинити симуляцію")
            self.run_simulation()
        else:
            self.simulation_running = False
            self.start_button.config(text="Почати симуляцію")

    def toggle_attack(self):
        self.attack_mode = not self.attack_mode
        if self.attack_mode:
            self.attack_button.config(text="Відмінити атаку")
        else:
            self.attack_button.config(text="Кібератака")

    def append_log(self, message, color="black"):
        self.log_area.config(state='normal')  # Розблокувати для редагування
        self.log_area.tag_configure(color, foreground=color)
        self.log_area.insert("1.0", message + "\n", color)
        self.log_area.config(state='disabled')  # Заблокувати для редагування

    def run_simulation(self):
        if not self.simulation_running:
            return

        if self.attack_mode:
            stream_data = self.autoencoder.create_stream_data_array(1, 1)
            temperature = stream_data[0][0]
            pressure = stream_data[0][1]
            water_level = stream_data[0][2]
        else:
            stream_data = self.autoencoder.create_stream_data_array(1, 0)
            temperature = stream_data[0][0]
            pressure = stream_data[0][1]
            water_level = stream_data[0][2]

        input_data = [temperature, pressure, water_level]

        result = self.autoencoder.analyze(input_data)

        self.temp_var.set(f"{temperature:.2f} °C")
        self.pressure_var.set(f"{pressure:.2f} kPa")
        self.water_var.set(f"{water_level:.2f} %")

        self.error_var.set(f"{result['mse']:.5f}")
        self.status_var.set(result['status'])

        if result['status'] == "АНОМАЛІЯ!":
            self.status_label.config(foreground="red")
            log_event(f"[LOG][{datetime.now()}] АНОМАЛІЯ! Вхідні дані: {input_data}, Відновлені дані: {result['reconstructed']}, MSE: {result['mse']:.5f}")
            # append log with red color
            self.append_log(f"[LOG][{datetime.now()}] АНОМАЛІЯ! Вхідні дані: {input_data}, Відновлені дані: {result['reconstructed']}, MSE: {result['mse']:.5f}\n", color="red")

        else:
```

```
            self.status_label.config(foreground="green")
            log_event(f"[LOG][{datetime.now()}] Норма. Вхідні дані: {input_data}, Відновлені дані: {result['reconstructed']}, MSE: {result['mse']:.5f}")
            self.append_log(f"[LOG][{datetime.now()}] Норма. Вхідні дані: {input_data}, Відновлені дані: {result['reconstructed']}, MSE: {result['mse']:.5f}\n")
        self.root.after(1000, self.run_simulation)

# main.py

from gui import MonitoringApp
import tkinter as tk

def main():
    root = tk.Tk()
    app = MonitoringApp(root)
    root.mainloop()

if __name__ == "__main__":
    main()

def log_event(message):
    with open("events.log", "a", encoding="utf-8") as f:
        f.write(f"{message}\n")

def load_events():
    try:
        with open("events.log", "r", encoding="utf-8") as f:
            lines = f.readlines()
        return lines
    except FileNotFoundError:
        return []
```

Додаток В. Ілюстративний матеріал

Розробка програмного засобу для виявлення атак на промислові системи керування (ICS) з використанням автоенкодерів

•••

Виконав ст. групи 1КІТС-21б
Керівник д. ф. (PhD), доц. кафедри МБІС

Актуальність:

Актуальність роботи полягає в необхідності створення спеціалізованих засобів виявлення кіберзагроз для промислових систем керування (ICS), з огляду на їхню зростаючу вразливість та критичне значення для інфраструктури.

Мета:

Метою роботи є розробка програмного засобу для виявлення аномальної активності у промислових системах керування ICS із використанням автоенкодерів, що дозволяє своєчасно виявляти потенційні загрози без наявності прикладів атак.

Завдання:

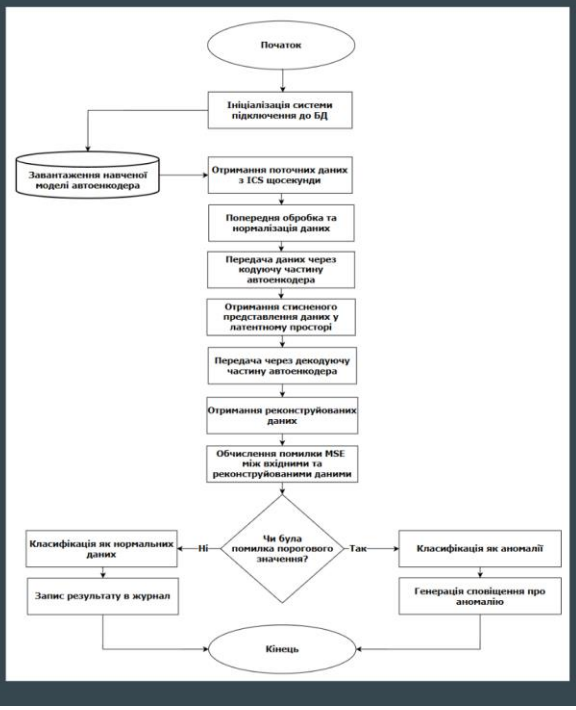
Завданням дослідження є аналіз особливостей роботи та вимог до кібербезпеки ICS, оцінка існуючих методів виявлення аномалій, розробка підходу на основі автоенкодерів, проєктування та реалізація програмного засобу для виявлення аномалій, його тестування на відкритому датасеті та підготовка користувацької інструкції.

Промислові системи керування (Industrial Control Systems, ICS) являють собою сукупність апаратного та програмного забезпечення, що використовується для автоматизації технологічних процесів на підприємствах і об’єктах критичної інфраструктури. В умовах швидкої цифровізації зростає актуальність забезпечення їхньої кібербезпеки, адже ці системи інтегруються з корпоративними мережами та хмарними платформами, що значно підвищує ризик кібератак.

Архітектура ICS включає SCADA-системи для цілісного контролю виробничих об'єктів, розподілені системи управління (DCS) для координації складних технологічних ланцюгів, інтерфейси "людина-машина" (HMI) для взаємодії персоналу з обладнанням, та спеціалізовані промислові мережі для надійного обміну даними між рівнями управління.

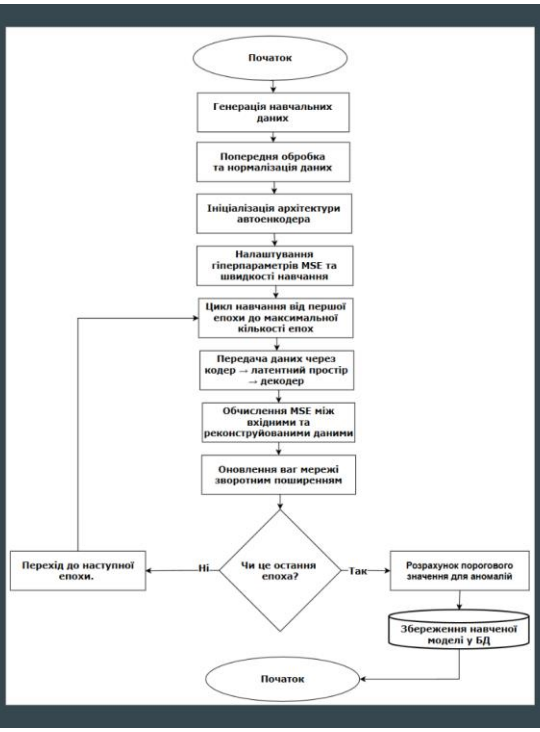
Автоенкодер — це тип штучної нейронної мережі, який використовується для навчання ефективного стислого подання (репрезентації) вхідних даних без нагляду. Він складається з двох основних частин: енкодера, який перетворює вхідні дані у стиснутий (прихований) простір, та декодера, який намагається відновити оригінальні дані з цього подання.

Метою автоенкодера є мінімізація різниці між початковими та відновленими даними, що дозволяє йому виявляти відхилення — зокрема, аномалії, які відрізняються від типових шаблонів у даних.



На даному зображенні наведено блок-схему, що відображає алгоритм роботи запропонованого підходу, що лежить в основі програмного забезпечення:

1) Починається процес виявлення аномалій
2) Встановлення з'єднання з базою даних
3) Завантаження навченої моделі автоенкодера
4) Збір телеметричних даних з промислової системи керування
5) Отримані сирі дані проходять етап попередньої обробки
6) Нормалізовані дані подаються на вхід кодуючої частини автоенкодера
7) Формується стиснене представлення вхідних даних
8) Стиснене представлення подається на вхід декодуючої частини автоенкодера
9) Отримуються реконструйовані значення технологічних параметрів
10) Розраховується середньоквадратична похибка
11) Розрахована помилка MSE порівнюється з попередньо встановленим пороговим значенням
12a) Класифікація як нормальних даних
12b) Класифікація як аномалії
13a) У базу даних записується інформація про поточний стан системи
13b) Система генерує сповіщення для операторів або систем автоматичного реагування
14) Запис результату та деталей аномалії у журнал БД
15) Процес виявлення аномалій для поточного набору даних завершено

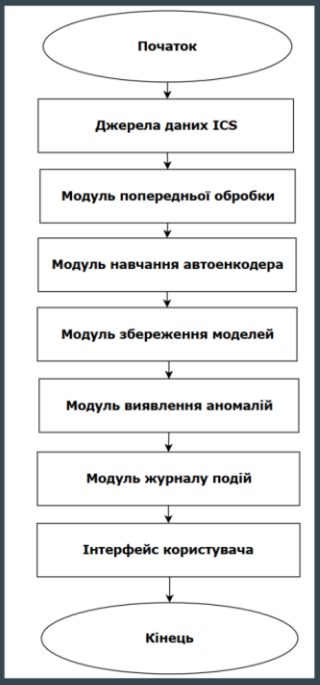


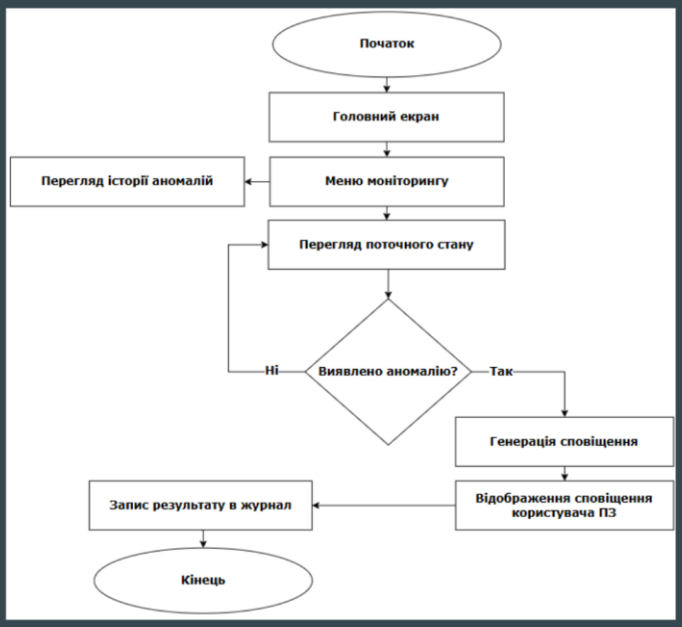
На данному зображенні відображено блок-схему алгоритму навчання автоенкодера:

1) Ініціюється процес створення моделі автоенкодера для виявлення аномалій в ICS.
2) Готуються дані нормального функціонування для навчання моделі.
3) Дані очищаються, нормалізуються та готуються до навчання.
4) Визначається структура автоенкодера з кодувальником, латентним простором і декодувальником.
5) Задаються основні параметри навчання, включаючи MSE і швидкість навчання.
6) Запускається багатоепоховий процес оптимізації моделі.
7) Дані проходять через кодер, латентний простір і декодер.
8) Визначається похибка між вхідними та реконструйованими даними.
9) Мережа навчається шляхом зворотного поширення помилки.
10) Перевіряється, чи досягнуто кінця навчання.
11) Якщо навчання триває, епоха збільшується і цикл повторюється.
12) Визначається поріг для виявлення аномалій на основі похибок.
13) Навчена модель і поріг зберігаються в базі даних.
14) Завершується навчання, модель готова до виявлення аномалій у реальному часі.

На даному зображенні наведено схему взаємодії компонентів архітектури розробленого ПЗ:

1) Запускається процес роботи програмного засобу.
2) Система отримує технологічні дані з промислових систем керування через підтримувані протоколи.
3) Дані очищуються та перетворюються у формат, придатний для подальшого аналізу.
4) На підготовлених даних виконується навчання автоенкодера для моделювання нормальної поведінки.
5) Навчена модель зберігається для подальшого використання у виявленні аномалій.
6) Модель аналізує поточні дані та виявляє аномалії, що можуть вказувати на атаки.
7) Виявлені аномалії реєструються в журналі подій із супровідною інформацією.
8) Користувач отримує дані про аномалії через зручний графічний інтерфейс.
9) Процес завершується, але система продовжує циклічний моніторинг ICS.





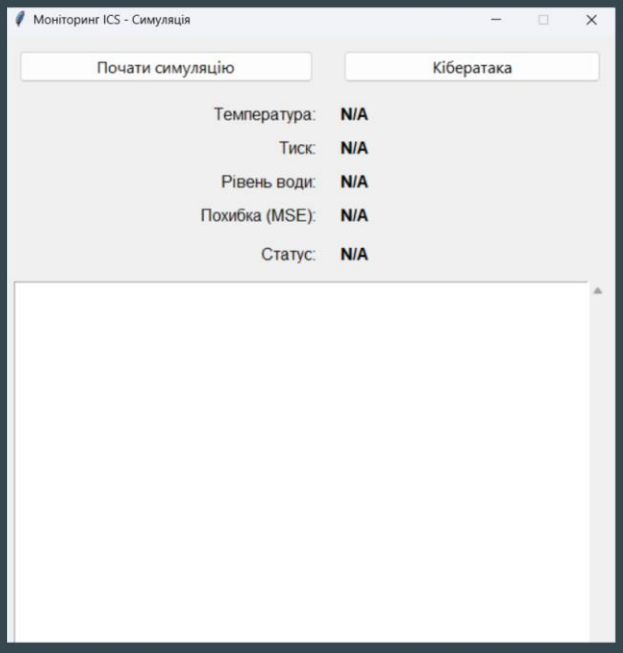
На даному зображенні наведено схема взаємодії користувача з функціоналом розробленого ПЗ:

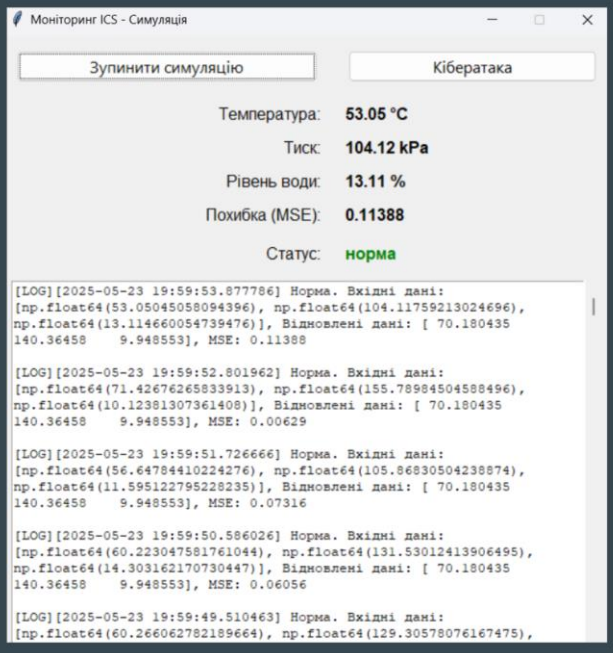
1) Користувач запускає програмний засіб і розпочинає роботу з системою.
2) Відкривається головне меню з основними функціями та елементами керування.
3) Користувач переходить до модуля моніторингу стану ICS.
4) Доступна історія виявлених раніше аномалій для аналізу.
5) Відображаються поточні параметри ICS та результати їх аналізу.
6) Система аналізує дані та визначає наявність або відсутність аномалій.
7а) У разі виявлення аномалії система формує відповідне сповіщення.
7б) Система продовжує спостереження без сповіщень.
8а) Користувачу показується повідомлення про виявлене відхилення.
9) Усі результати аналізу автоматично фіксуються в журналі подій.
10) Завершується поточний сеанс взаємодії користувача з функціоналом моніторингу.

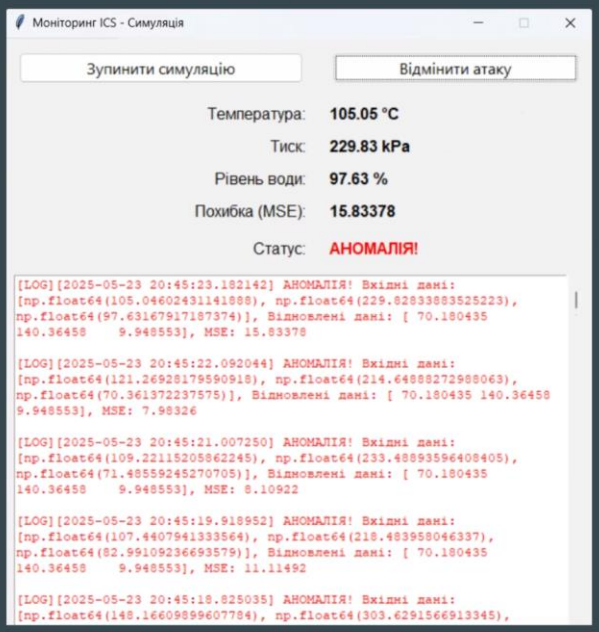
У процесі вибору розглядалися такі популярні мови програмування, як Python, C++, Java та C#. Основними критеріями оцінки були: підтримка глибокого навчання, зручність реалізації, наявність відповідних бібліотек, а також можливість швидкого створення прототипу.

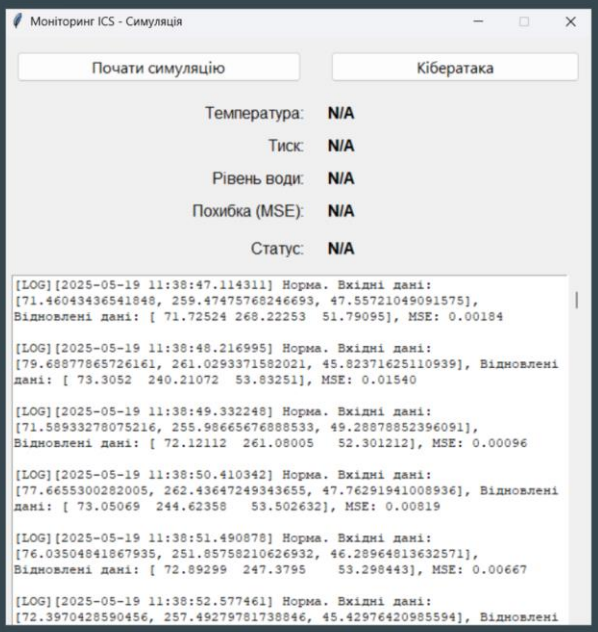
Python є безумовним лідером у сфері машинного навчання. Завдяки таким бібліотекам, як TensorFlow, Keras, PyTorch, а також наявності високорівневих API, Python дозволяє швидко реалізовувати та тестувати моделі глибокого навчання, зокрема автоенкодери. Крім того, Python підтримує створення GUI-додатків за допомогою бібліотек Tkinter, PyQt або Kivy, що є зручним для реалізації користувацького інтерфейсу без необхідності застосування складних фреймворків.









Результат роботи

У результаті виконання бакалаврської кваліфікаційної роботи було успішно реалізовано програмний засіб для виявлення атак на промислові системи керування (ICS) з використанням автоенкодера — нейронної мережі, здатної навчатися на "нормальних" даних та виявляти відхилення у режимі реального часу. Запропонований підхід дозволяє не лише виявляти вже відомі типи атак, але й ефективно ідентифікувати нові, раніше невідомі загрози, які не можуть бути розпізнані традиційними сигнатурними методами. Система демонструє високу чутливість до аномалій, що дає змогу своєчасно реагувати на потенційні інциденти безпеки без втручання в штатну роботу обладнання. Реалізований інтерфейс користувача забезпечує зручний доступ до інформації про стан системи та дозволяє аналізувати як поточні, так і історичні дані. Таким чином, розроблений програмний засіб сприяє суттєвому підвищенню рівня кіберзахисту критичної інфраструктури та забезпечує надійний моніторинг промислових процесів у сучасних умовах цифровізації.

Дякую за увагу

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ Кафедра менеджменту на безпеки інформаційних систем
факультет менеджменту та інформаційної безпеки
Гр. 1КІТС-21б

Керівник доцент Салієва О.В. _____

Показники звіту подібності

Strike Plagiarism	
Оригінальність	99,72 %
Загальна схожість	0,28 %

Аналіз звіту подібності (відмітити потрібне)

- ☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Підлуцький О.А.
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

- ☐ **Допустити до захисту**

_____

_____

_____

_____

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

_____

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)