Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-25М
ОПП «Комп’ютерна інженерія»
спеціальності F7 «Комп’ютерна інженерія»
_____ Кваша М.В.
«____» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Буравченко К.О.
«____» _____ 2026 р.

Нормоконтроль проекту
_____ Коваленко А.С.
«____» _____ 2026 р.

Рецензент _____
_____

Підписано КЕП
Центральноукраїнський національний технічний університет
Кваша Михайло Віталійович
Сертифікат: 18DF2C9B93CEA20404000000494915570001F64406
Дійсний з 08.10.2026 22:17:00 по 08.10.2027 22:16:59

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _F_____*"*Інформаційні технології*"*_____

Спеціальність F7 *"*Комп'ютерна інженерія*"*

Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 1 » вересня 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Кваші Михайлу Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах*

2. Керівник роботи *Буравченко Костянтин Олегович, кандидат технічних наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 41-13 від 17.08.2026 року

3. Строк подання студентом роботи до захисту *2.12.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*5. Впровадження системи в промислову експлуатацію*
2. Перегляд аналогічних існуючих систем.	*6. Наукова новизна*.
3. Опис і обґрунтування проектних рішень.	*7. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*8. Висновки.*

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Марченко К.М., к.т.н., доцент	06.10.2026	16.11.2026

8. Дата видачі завдання « 1 » вересня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2026 р.	
3.	Розробка моделі компонента	20.10.2026 р.	
4.	Розробка структур даних	25.10.2026 р.	
5.	Розробка алгоритмів системи	30.10.2026 р.	
6.	Програмування алгоритмів	10.11.2026 р.	
7.	Розрахунки з охорони праці та техніки безпеки	15.11.2026 р.	
8.	Оформлення ПЗ	17.11.2026 р.	
9.	Попередній захист роботи	2.12.2026 р.	

Дата видачі завдання
« 1 » вересня 2026 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 1 » вересня 2026 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Кваша М.В. Дослідження та програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах. F7 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Метою розробки є дослідження та програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Об'єктом дослідження є процес моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Предметом дослідження є методи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Методи дослідження базуються на методах побудови комп'ютерних мереж, методів теорії харних обчислень, методів теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, моніторинг та оцінка вразливостей додатків, хмарні обчислювальні середовища

ABSTRACT

Kvasha M.V. Research and Software Implementation of a System for Monitoring and Assessing Application Vulnerabilities in Serverless Cloud Computing Environments. F7 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

This master's-level final qualification thesis presents the development of software designed for a system to monitor and assess application vulnerabilities within serverless cloud computing environments.

The objective of this project is to research and implement a software system for monitoring and assessing application vulnerabilities in serverless cloud computing environments.

The object of the research is the process of monitoring and assessing application vulnerabilities in serverless cloud computing environments.

The subject of the research consists of methods for monitoring and assessing application vulnerabilities in serverless cloud computing environments.

The research methodology draws upon techniques for computer network construction, cloud computing theory, information security theory, mathematical statistics, and software development.

The outcome of the work is a software implementation of a system for monitoring and assessing application vulnerabilities in serverless cloud computing environments.

During the development of the software model, an analysis of existing hardware and software tools was conducted. All components of the developed software are described in detail.

A user-friendly interface has been developed. Instructions for using the software tools are provided.

The program can be used on PCs running the Windows 10/11 operating system.

The program was developed in a Python environment.

Keywords: computer engineering, application vulnerability monitoring and assessment, cloud computing environments

ВІДОМІСТЬ ПРОЕКТУ

		Текстові документи			
1	А4	Пояснювальна записка	99	–	
2	А4	Робоча програма	28	–	Додаток А
		Графічні документи			
3	А4	Наукова новизна	1	–	
4	А4	Структурна схема	1	–	
5	А4	Функціональна схема роботи системи	1	–	
6	А4	Діаграма взаємодії процесів	1	–	
7	А4	Блок-схема основної програми	1	–	
8	А4	Блок-схема роботи підпрограми	1	–	

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 9
ВСТУП..... 10
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 12
1.1 Призначення системи..... 12
1.2 Область застосування 13
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 16
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 16
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 32
2.3 Розгорнута постановка завдання 33
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 35
3.1 Опис функціонування системи 35
3.2 Розробка структурної схеми..... 38
3.3 Розробка функціональної схеми 40
3.4 Розробка діаграми процесів 49
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 50
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 50
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 73
6 НАУКОВА НОВИЗНА 79

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 80
7.1 Вступ 80
7.2 Шкідливі і небезпечні фактори при роботі з комп’ютером 81
7.3 Розробка заходів з умов поліпшення охорони праці 82
7.4 Пожежна безпека 83
7.5 Розрахункова частина 86
8 ОСНОВНІ ВИСНОВКИ 90
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 92

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ДВЧ	–	датчик випадкових чисел
ДСТ 28147-89	–	алгоритм шифрування
ЕОМ	–	електронна обчислювальна машина
ІС	–	інформаційна система
ОС	–	обчислювальна система
ПВЧ	–	псевдовипадкові числа
ПЗ	–	програмне забезпечення
РПЗ	–	руйнуючі програмні засоби
СЗІ	–	система захисту інформації
ASCII	–	система кодування
DES	–	алгоритм шифрування
FEAL	–	алгоритм шифрування
IDEA	–	алгоритм шифрування
KOI-8	–	система кодування
RISC	–	архітектура процесора

ВСТУП

Актуальність теми. Інструменти оцінки безпеки хмари перевіряють хмарну інфраструктуру, сервіси, ідентифікаційні дані, робочі навантаження та стани конфігурації на відповідність політикам безпеки, системам відповідності та вимогам внутрішнього контролю.

Вони допомагають командам знаходити незахищені активи, слабкі шляхи ідентифікації, відсутні журнали, прогалини в шифруванні, вразливі робочі навантаження, ризиковані мережеві правила, відхилення політик та збої в контролі аудиту.

У старіших програмах оцінювання результатом часто був звіт. У сучасних корпоративних середовищах результат має стати операційним робочим процесом: у 2026 році корисним розмежуванням не є «чи сканує інструмент?». Майже кожен інструмент сканує. Краще питання полягає в тому, чи може платформа пояснити, чому результат має значення в реальному середовищі, і довести, що сталося після створення результату.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

– Дослідження системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

– Програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Об’єктом дослідження є процес моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Предметом дослідження є методи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Методи дослідження базуються на методах побудови комп'ютерних мереж, методів теорії харних обчислень, методів теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

– Розроблено вітчизняний продукт моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVIII науково-технічній конференції здобувачів вищої освіти «Наука-виробництву», основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №17.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Інструменти оцінки безпеки хмарних технологій перетинаються з кількома категоріями безпеки хмарних технологій, оскільки оцінка – це робочий процес, а не окремий тип продукту. Одна й та сама платформа може включати можливості, пов’язані з CSPM, CNAPP, CIEM або SIEM, залежно від того, як вона виявляє ризики, визначає пріоритети результатів та підтримує виправлення:

– CSPM. Управління безпекою хмарних технологій виявляє неправильні конфігурації хмарних технологій та прогалини в роботі облікових записів, підписок, проектів та служб. Це основний рівень оцінювання для конфігурації хмарних технологій, перевірок контрольних показників та контролю відповідності.

– CNAPP. Платформа захисту додатків, що базується на хмарних технологіях, поєднує в одній платформі безпеку стану, робочого навантаження, ідентифікації, вразливостей, IaC, контейнерів та середовища виконання. Це ширша категорія, яка часто включає функції оцінки безпеки хмари, особливо коли інструмент пов'язує неправильні конфігурації, вплив, вразливості та шляхи ідентифікації.

– CIEM. Управління правами доступу до хмарної інфраструктури аналізує хмарні ідентифікаційні дані, дозволи, права доступу, ролі, облікові записи служб та шляхи привілеїв. Це важливо для оцінки, оскільки надмірний доступ та нелюдські ідентифікаційні дані зараз є основним джерелом хмарних ризиків.

– CWPP. Платформа захисту хмарних робочих навантажень (Cloud Workload Protection Platform) захищає такі робочі навантаження, як віртуальні машини, контейнери та серверні навантаження. Вона додає контекст середовища виконання та робочого навантаження до результатів оцінювання, щоб команди

могли зрозуміти, чи впливає проблема стану на активне робоче навантаження, чи це просто ще один покинутий ресурс, володіння яким ніхто не визнає.

– KSPM. Kubernetes Security Posture Management перевіряє конфігурацію Kubernetes, робочі навантаження, кластери, RBAC, мережеві політики та налаштування площини керування. Це необхідно, коли оцінка хмари включає керовані служби Kubernetes, такі як EKS, AKS та GKE.

– DSPM. Управління безпекою даних виявляє витоки конфіденційних даних, ризиковані шляхи доступу та неправильні конфігурації сховищ даних. Це корисно, коли оцінка має показати, де зберігаються конфіденційні дані, хто може до них отримати доступ і чи дійсно працюють навколишні засоби контролю.

– SIEM. Управління інформацією та подіями безпеки (SIEM) зіставляє журнали, сповіщення та події безпеки в різних системах. Зазвичай це не сам по собі інструмент оцінки безпеки хмари, але він часто отримує сигнали оцінки для робочих процесів SOC, розслідування та реагування на інциденти.

1.2 Область застосування

Інструменти оцінки безпеки хмари постійно перевіряють хмарні середовища. Послуги оцінки безпеки хмари – це експертні заняття, під час яких консультанти переглядають архітектуру, конфігурацію, політики, ризики та пріоритети виправлення.

Більшості корпоративних команд потрібні обидва, але не для однієї й тієї ж роботи. Сервіс може сказати вам, що операційна модель слабка. Інструмент повинен постійно доводити, що операційна модель працює після того, як консультанти підуть, і хмарна інфраструктура одразу ж знову почне змінюватися, тому що саме це робить інфраструктура, коли за нею ніхто не стежить.

Типи інструментів оцінки безпеки хмари

«Оцінка безпеки хмари» може означати сканування конфігурації, повне розгортання CNAPP, робочий процес підтвердження відповідності або конвеєр збагачення SOC. Ми зосереджуємося на інструментах, які оцінюють стан хмари в реальному часі та допомагають командам визначати пріоритети та направляти результати.

1. Комерційні платформи оцінки безпеки хмарних технологій

Комерційні платформи оцінки безпеки хмарних технологій зазвичай краще підходять, коли програмі потрібні періодичні сканування, нормалізована інвентаризація активів, звітність про відповідність вимогам, маршрутизація робочих процесів, підтримка та результати, зрозумілі для керівництва.

Покупці зазвичай змінюють інструменти, тому що стара платформа зазнала невдачі в одному з чотирьох пунктів: слабкий контекст, слабка пріоритизація, слабка маршрутизація власності або слабкі докази.

2. Інструменти оцінки безпеки хмарних технологій з відкритим кодом

Інструменти з відкритим кодом все ще важливі для перевірок, керованих інженерами, тактичних аудитів, перевірки CI/CD та прозорої логіки правил.

Зазвичай вони не є повноцінною заміною комерційних платформ у корпоративних середовищах, оскільки робочий процес обробки доказів, зіставлення власності, життєвий цикл винятків, підтримка та нормалізація кількох хмар все ще потребують внутрішньої розробки або підтримки.

Практичний поділ полягає у використанні інструментів з відкритим кодом для інженерної валідації та комерційних платформ для безперервної оцінки, звітності, володіння та робочого процесу аудиту.

Ринок послуг оцінки безпеки хмарних технологій у 2026 році

Ринок послуг оцінки хмарної безпеки відходить від перевірок стану в певний момент часу до безперервних робочих процесів оцінки. Послуги все ще важливі для перевірки архітектури, готовності до аудиту, роботи із запобігання порушенням та перевірки на рівні ради директорів, але рівень контролю все більше керується програмним забезпеченням.

Кілька ринкових сигналів пояснюють, чому покупці оцінюють інструменти та послуги разом у 2026 році.

1. CNAPP – це сегмент, що найшвидше розвивається, з темпами зростання майже 14,6%, оскільки компанії відходять від фрагментарних інструментів безпеки.

2. Оцінювання хмарних технологій стає частиною щоденних операцій з безпеки, а не щоквартальним артефактом аудиту, який хтось експортує, перш ніж відділ відповідності поставить незручні запитання.

3. Очікується, що сервіси стануть найшвидше зростаючим сегментом ринку хмарної безпеки, що зумовлено складністю хмарних технологій, постійним моніторингом, виявленням загроз, консалтингом та управлінням відповідністю вимогам. Інструменти все ще потребують перевірки людиною, але сканування та збір доказів повинні виконуватися безперервно.

4. 81% компаній зазнали щонайменше одного інциденту безпеки хмари за останній рік. Тому інструменти оцінювання повинні виявляти незахищені активи, слабкі шляхи ідентифікації, неправильні конфігурації та вразливі робочі навантаження, перш ніж вони стануть роботою реагування на інциденти.

5. Підприємства використовують у середньому понад 35 інструментів моніторингу безпеки, що створює прогалини у видимості у фрагментованих багатохмарних середовищах.

Це створює різний тиск покупців залежно від типу команди: IT-лідам малого та середнього бізнесу потрібен швидкий адаптаційний процес та чітке ціноутворення; Універсалам з хмарних операцій потрібні інформаційні панелі та пріоритети; Молодшим інженерам з безпеки потрібні конкретні висновки та рекомендації щодо усунення недоліків; Корпоративним командам потрібні докази, відповідальність, інтеграція та обробка винятків

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Як вибрати інструмент для оцінки безпеки хмарних технологій

Почніть з режиму несправності, який ви намагаєтеся виправити. Якщо поточний інструмент вже сканує середовище, не купуйте інший сканер із чистішою демо-версією.

Перш ніж порівнювати постачальників, перевірте ці критерії:

– Хмарне покриття. Інструмент має оцінювати середовища, які ви фактично використовуєте, а не середовища, які надає перевагу демонстрація постачальників. Для більшості команд це означає AWS, Azure, GCP, Kubernetes, а іноді й інвентаризацію Oracle Cloud, VMware або SaaS.

– Контекст активу. Результати повинні бути пов’язані з реальними активами, а не лише з ідентифікаторами хмарних ресурсів. Шукайте власника, програму, середовище, бізнес-одиницю, регіон, обліковий запис, підписку, проект, теги та контекст залежностей.

– Пріоритетність ризиків. Тільки серйозності недостатньо. Корисний інструмент повинен враховувати рівень ризику, шляхи ідентифікації, критичність робочого навантаження, відомі вразливості, доступність до Інтернету, вплив на виробництво, а також те, чи впливає проблема на реальний сервіс, чи просто на черговий покинутий ресурс, що тихо гниє в обліковому записі.

– Оцінка ідентифікаційних даних. Перевірте, чи може платформа оцінювати користувачів, ролі, облікові записи сервісів, дозволи, надмірні привілеї та нелюдські ідентичності. Це важливіше у 2026 році, оскільки автоматизація та робочі процеси штучного інтелекту продовжують додавати

шляхи доступу до машин, які ніхто не хоче перевіряти вручну, що шокує.

– Відповідність та докази. Інструмент повинен зіставляти результати з такими фреймворками, як CIS Benchmarks, NIST, PCI DSS, HIPAA, SOC 2, ISO 27001, ISO 27017, GDPR та FedRAMP, де це доречно. Логотипи фреймворків на сторінці з цінами не враховуються; попросіть надати докази на рівні контролю, варіанти експорту, історію заявок, схвалення та підтвердження виправлення.

– Робочий процес виправлення. Результати дослідження повинні бути спрямовані на робочі процеси системних інженерів, які вже використовують, такі як Jira, ServiceNow, PagerDuty, Slack, електронна пошта, SIEM або GRC. Панель інструментів, яку ніхто не відкриває, не є виправленням. Перевірте, чи включають винятки коди причин, терміни дії, осіб, що затверджують, уражену область та історію переглядів, замість того, щоб стати постійним сховищем для ризиків.

– Вартість переходу. Якщо ви замінюєте існуючий інструмент, безпосередньо протестуйте тертя міграції: інтеграції, існуючі заявки, звіти про відповідність, історію винятків, зіставлення політик, дані про власність, модель ціноутворення та перегляньте відгуки від G2, Capterra, TrustRadius або Gartner Peer Insights.

Cloudaware

Найкраще підходить для: команд із забезпечення безпеки, відповідності та платформ, що працюють у кількох хмарах, яким потрібні результати оцінки хмарних ресурсів, пов’язані з контекстом CMDB, правом власності, робочим процесом виправлення та аудиторськими доказами.

Cloudaware збагачує оцінку безпеки хмарних ресурсів контекстом CMDB. Він виявляє хмарні, локальні та гібридні активи, нормалізує їх, оцінює їх за понад 550 попередньо створеними політиками безпеки та відповідності, а також пов’язує результати з правом власності, робочими процесами виправлення та аудиторськими доказами.

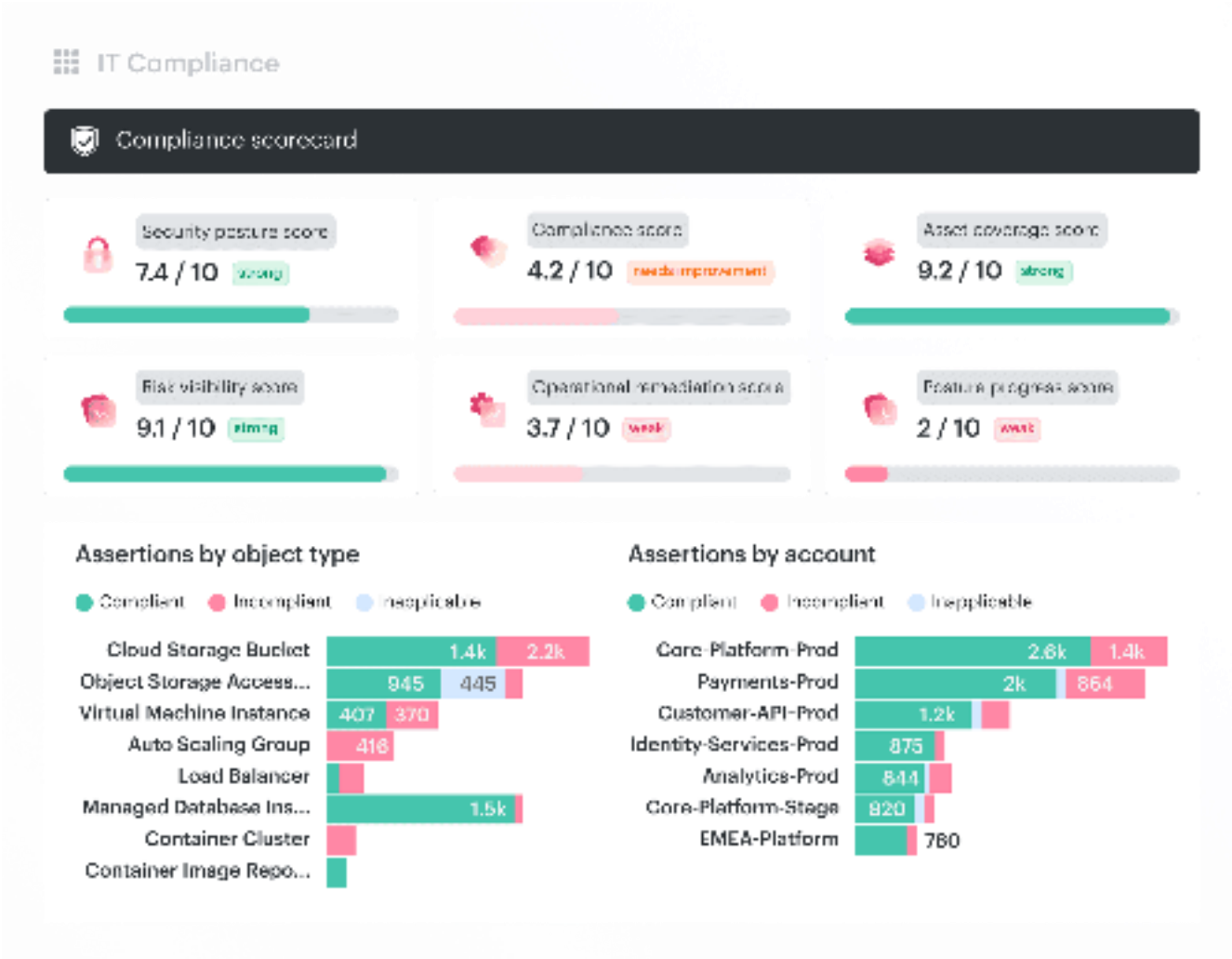


Рисунок 2.1 – Інтерфейс користувача Cloudaware

Це підходить для команд, які оцінюють хмарні, локальні та гібридні активи, де результати оцінки стану не можуть залишатися ізольованими від контексту інвентаризації, власності та відповідності. Платформа підтримує робочі процеси CSPM, управління вразливостями, збагачення журналів SIEM та відповідність IT-стандартам, тому результати оцінювання можуть бути перенесені в заявки, докази, інформаційні панелі та аудиторські перевірки.

Інструменти оцінювання в Cloudaware:

– Багатохмарна CMDB: автоматично виявляє та нормалізує активи в AWS, Azure, GCP, Oracle, Alibaba, VMware, SaaS та підключених середовищах, надаючи результатам оцінювання реальний рівень активів, а не лише ідентифікатори ресурсів, надані постачальником.

– Політики CSPM: Оцінює засоби керування безпекою хмари, використовуючи детальні дані конфігурації CMDB, щоб результати можна було пов’язати з хмарними об’єктами, обліковими записами, підписками, проектами, власниками та середовищами.

– CIS Benchmarks: Підтримує оцінювання на основі бенчмарків за допомогою документації механізму відповідності Cloudaware, включаючи спеціальний розділ CIS Benchmarks.

– Запити на користувацькі політики: дозволяє командам запитувати користувацькі політики механізму відповідності з деталями політики, логікою оцінювання, деталями порушення та тестовими об'єктами.

– Робочі процеси управління вразливостями: Контекстуалізуйте вразливості за допомогою даних CMDB, допомагаючи командам призначати виявлені вразливості відповідному власнику, ініціативі або завданню усунення, а не розглядати CVE як плоский список.

– Збагачення SIEM: покращує журнали хмарних служб атрибутами CMDB, що допомагає командам SOC досліджувати події з урахуванням контексту активів, власників та середовищ.

– Збір доказів: IT Compliance автоматизує збір та оцінку доказів, зменшуючи ручну роботу з відновлення скріншотів, експорту та доказів перед аудитами.

– Маршрутизація виправлення: Cloudaware підтримує операційну передачу відповідальності через такі інтеграції, як Jira та PagerDuty, роблячи результати оцінювання частиною існуючих інженерних робочих процесів.

– Обробка винятків: Використовуйте робочі процеси політик та порушень Cloudaware, щоб відокремити прийняті або виходять за межі області дії результати від активних черг виправлення.

Ціноутворення на хмарне програмне забезпечення

Cloudaware пропонує 30-денну безкоштовну пробну версію та загальнодоступний калькулятор цін. На сторінці цін показано початковий приклад ціни $200/місяць, включаючи ємність до 25 000 елементів конфігурації,

з оцінками, починаючи з 50 серверів. Калькулятор рентабельності інвестицій дозволяє покупцям моделювати прогнозовану економію, використовуючи такі вхідні дані, як розмір організації, обсяг активів, обсяг заявок, область відповідності та вибрані модулі, включаючи CMDB, управління вразливостями, управління програмними активами, FinOps та CSPM.

Припиніть перебудову доказів безпеки хмари перед кожним аудитом. Cloudaware оцінює активи на відповідність понад 550 політикам безпеки та відповідності, пов’язує порушення з власниками та робочими процесами усунення порушень, а також зберігає докази, пов’язані із записом активу.

Wiz

Найкраще підходить для: оцінки CNAPP без агента та видимості графіка хмарних ризиків.

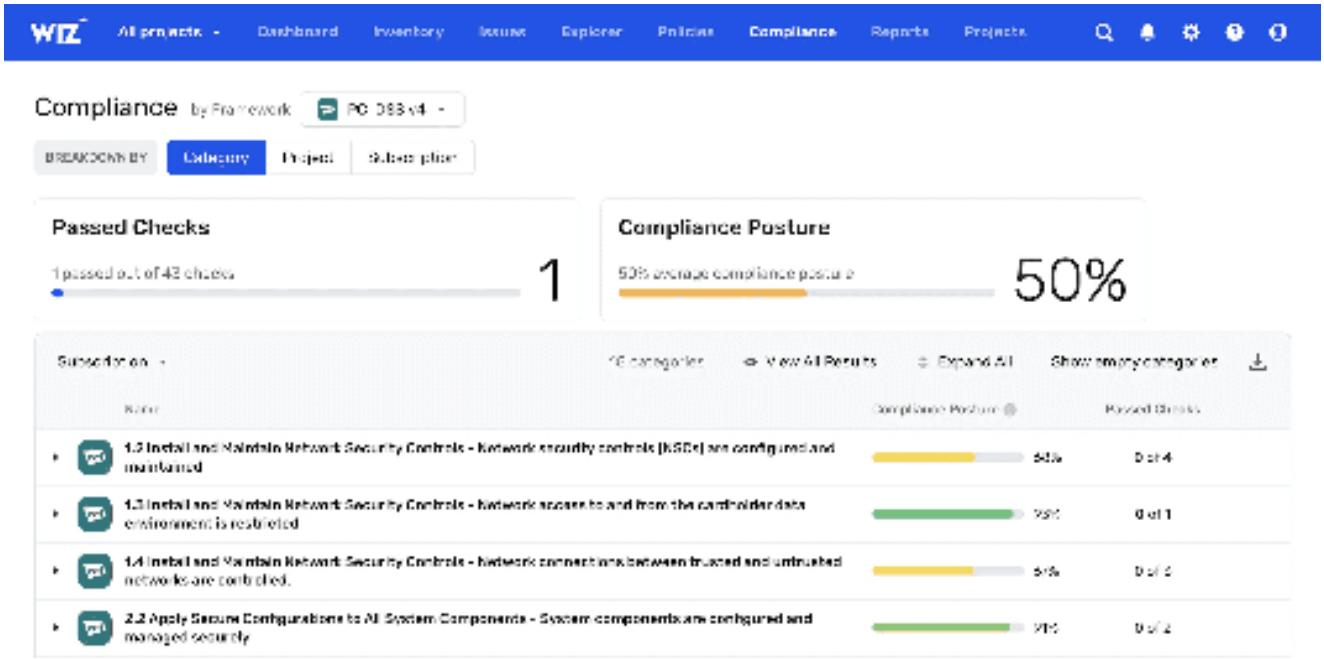


Рисунок 2.2 – Інтерфейс користувача Wiz

Wiz побудовано на основі безагентного виявлення та пріоритезації ризиків на основі графів. Він найефективніший, коли команді потрібно об'єднати неправильні конфігурації, вразливості, публічний доступ, шляхи ідентифікації,

конфіденційні дані та шляхи атак в одне хмарне подання ризиків.

Для покупців оцінювання Wiz добре підходить, коли швидке впровадження та широка видимість CNAPP важливіші, ніж глибоке володіння CMDB або робочі процеси, що підтверджують аудит.

Функції Wiz:

– Безагентне сканування: підключається через API-конектори до хмарних середовищ інвентаризації та середовищ штучного інтелекту без розгортання агентів всюди.

– CSPM/CNAPP: Охоплює стан хмари, управління вразливостями, CIEM, безпеку контейнерів та Kubernetes, DSPM, відповідність вимогам, захист хмарних робочих навантажень, сканування IaC та безпеку штучного інтелекту.

– Аналіз шляхів атаки: використовує Wiz Security Graph, щоб показати, як вразливості, ідентифікаційні дані, вразливості та конфігурації поєднуються у шляхи порушення.

– Пріоритизація вразливостей: додає хмарний контекст до вразливостей у віртуальних машинах, безсерверних системах, контейнерах та пристроях.

– Контекст ідентифікації та даних: Включає охоплення CIEM та DSPM для дозволів, ризику, пов’язаного з правами доступу, конфіденційних даних та витоку даних.

Ціни Wiz

Ціни Wiz не публікують фіксованої ціни. На сторінці цін використовується процес запиту, і зазначено, що ліцензування є модульним, масштабується залежно від робочих навантажень, активних розробників, обробки журналів або датчиків. Wiz перелічує окремі області ліцензування, такі як Wiz Cloud, Wiz Code, Wiz Defend, Wiz Sensor та Wiz Go Bundle для малого та середнього бізнесу.

CrowdStrike Falcon Cloud Security

Найкраще підходить для: команд, яким потрібна оцінка хмарних технологій, пов'язана з робочим навантаженням, часом виконання та виявленням загроз.

CrowdStrike Falcon Cloud Security є найсильнішим рішенням, коли оцінка хмарних технологій має бути тісно пов'язана із захистом середовища виконання та робочими процесами SecOps. Це не просто інструмент для оцінки стану; він знаходиться всередині ширшої платформи Falcon, де дані про хмарні дані можуть бути пов'язані із захистом робочого навантаження, безпекою контейнерів, станом Kubernetes, пріоритезацією вразливостей та виявленням загроз.

Це краще підходить для команд, які вже стандартизовані на CrowdStrike, або для команд з безпеки, які не хочуть, щоб дані про стан тіла були відокремлені від сигналів під час виконання.

Функції безпеки CrowdStrike Falcon Cloud:

– CNAPP/CSPM: Охоплює стан хмари та ризики, пов'язані з хмарними застосунками.

– Захист хмарного робочого навантаження: пов'язує результати оцінювання із захистом робочого навантаження під час виконання.

– Безпека контейнерів та Kubernetes: Підтримує варіанти використання контейнерів, образів та Kubernetes.

– Пріоритезація вразливостей: визначення пріоритетів хмарних ризиків з використанням вразливості та контексту зловмисника.

– Виявлення загроз: Переносить виявлення та реагування на хмарні загрози на платформу Falcon.

Ціни на CrowdStrike Falcon Cloud Security

Ціни на CrowdStrike залежать від модулів Falcon, покриття хмарного робочого навантаження, захисту середовища виконання та розміру розгортання.

Orca Security

Найкраще підходить для: оцінки безпеки хмари без агентів із чіткою пріоритезацією ризиків.

Orca Security – це безагентна платформа CNAPP для команд, яким потрібна широка хмарна видимість без розгортання агентів для кожного робочого навантаження. Її оціночна цінність є найбільш цінною в середовищах,

де необхідно одночасно пріоритезувати інвентаризацію, вразливість, ризики ідентифікації, контекст робочого навантаження, вразливості та результати відповідності вимогам.

Orca найбільш корисний, коли списки плоских сповіщень перестали працювати, і команді потрібен контекст шляху атаки, щоб вирішити, що слід виправити в першу чергу.

Функції безпеки Orca:

– Безагентне сканування: Використовує технологію Orca SideScanning для сканування хмарних ресурсів без попереднього розгортання агента.

– CSPM/CNAPP: Охоплює випадки використання, пов'язані з станом, робочим навантаженням, ідентифікацією, даними, вразливістю, API, відповідністю вимогам та безпекою застосунків.

– Шляхи атак: Виявляє ризики на основі хмарного контексту та впливу на бізнес.

– Пріоритетність вразливостей: Пріоритетність важливих ризиків замість того, щоб завалювати команди необробленими висновками.

– Охоплення контейнерів та Kubernetes: Включає безпеку контейнерів, образів та Kubernetes.

– Звітність про відповідність: Підтримка відповідності кількох хмарним системам та налаштування перевірок.

Ціни на Orca Security

Ціноутворення Orca залежить від хмарних ресурсів, робочих навантажень, облікових записів, увімкнених модулів або масштабу розгортання.

Згідно з Vendr, орієнтовні діапазони річних контрактів включають:

– Невеликі розгортання (500-2000 робочих навантажень): 50 000-150 000 доларів США на рік.

– Розгортання на середньому ринку (2000-10 000 робочих навантажень): 150 000-500 000 доларів США на рік.

– Розгортання підприємств (понад 10 000 робочих навантажень): від 500 000 доларів США на рік.

Tenable Cloud Security

Найкраще підходить для: команд з управління вразливістю, що поширюють програми оцінки вразливостей на хмарні технології.

Tenable Cloud Security підходить для команд, які вже працюють з управлінням вразливостями. Його цінність найбільша, коли потрібно аналізувати дані про стан хмарних технологій разом із вразливостями, ідентифікаторами, ризиками робочого навантаження, вразливістю конфіденційних даних та контекстом штучного інтелекту/хмарних активів.

Це особливо актуально, коли організація вже використовує Tenable для управління вразливостями та хоче, щоб оцінка безпеки хмари використовувалася для тієї ж моделі впливу.

Функції Tenable Cloud Security:

– CSPM: Виявляє активи, виявляє неправильні конфігурації та підтримує безперервне керування станом хмари.

– CIEM: Оцінює ризик ідентифікації хмари, шляхи доступу, найменші привілеї та постійні привілеї.

– Управління ризиками: співвідносить хмарні ризики, щоб показати токсичні комбінації між неправильними конфігураціями, вразливостями, ідентифікаторами та конфіденційними даними.

– Захист робочого навантаження: охоплює контейнери, Kubernetes, безсерверні системи та контекст вразливостей, пов'язаних з процесом збірки для виконання.

– DSPM: Виявляє конфіденційні дані та ресурси штучного інтелекту в багатохмарних середовищах.

Ціни на Tenable Cloud Security

Tenable не публікує фіксовану публічну ціну на Tenable Cloud Security. Наприклад, Tenable Vulnerability Management відображається як річна підписка від 3500 доларів США за 100 ресурсів в одному віджеті цін, тоді як в іншому розділі цін Tenable показано 3700 доларів США за максимум 250 ресурсів.

Управління хмарною безпекою Datadog

Найкраще підходить для: команд, які вже використовують Datadog для спостереження та бажають проводити оцінку хмарних даних в одній операційній консолі..

Управління безпекою хмарних систем Datadog є найефективнішим для команд, які вже використовують інфраструктуру, журнали, APM, моніторинг контейнерів, Kubernetes або робочі процеси інцидентів у Datadog. Його оціночна цінність полягає в тому, що сигнали стану, вразливості, відповідності, прав доступу, робочого навантаження та SIEM-суміжні сигнали розташовані поруч із тими, які вже використовують інженери телеметрії.

Це не стільки окрема платформа для підтвердження відповідності, скільки рівень операційної безпеки в середовищі з високим рівнем спостереження.

Функції управління хмарною безпекою Datadog:

– CSPM: Виявляє неправильні конфігурації та проблеми з хмарним станом.

– CIEM: Покриває ризики, пов’язані з правами доступу та дозволами, за допомогою набору продуктів хмарної безпеки Datadog.

– Управління вразливостями: пов'язує інфраструктуру та контекст робочого навантаження з вразливостями.

– Відповідність: Підтримує моніторинг відповідності та орієнтовані на еталони перевірки стану.

– Захист робочого навантаження: додає сигнали робочого навантаження під час виконання для контексту загроз та робочого навантаження.

Ціни на управління хмарною безпекою Datadog

Datadog публікує публічні ціни на послуги хмарної безпеки.

– CSM Pro починається від 10 доларів США за хостинг-провайдера на місяць при щорічній оплаті або від 12 доларів США за запитом.

– Вартість CSM Enterprise починається від 25 доларів США за хостинг-провайдера на місяць при щорічній оплаті або від 30 доларів США за запитом.

– Cloud SIEM має власну модель ціноутворення, що базується на

проаналізованих та отриманих або просканованих гігабайтах подій.

Check Point CloudGuard

Найкраще підходить для: оцінки стану хмари з урахуванням політики безпеки та управління мережевою безпекою.

Check Point CloudGuard краще підходить, коли оцінка хмарних технологій має пов’язати контроль стану, робочого навантаження, додатків та мережевої безпеки. Він входить до ширшого портфоліо хмарних рішень Check Point, орієнтованих на запобігання, тому він більш актуальний для команд, які хочуть пов’язати оцінку стану з правозастосуванням.

Це особливо корисно для організацій, які вже використовують засоби контролю безпеки Check Point, або для команд, які оцінюють хмарні технології разом із вимогами до WAF, брандмауера, запобігання загрозам та управління гібридною хмарою.

Функції Check Point CloudGuard:

– CSPM/CNAPP: Підтримує захист хмарних програм з орієнтацією на запобігання.

– Безпека хмарної мережі: Поєднує оцінювання з хмарними брандмауерами, віртуальною глобальною мережею та можливостями мережевої безпеки.

– Захист робочого навантаження: охоплює хмарні робочі навантаження в різних додатках та інфраструктурі.

– Перевірки відповідності: Підтримує оцінку стану та відповідності хмарних систем.

– Запобігання загрозам: додає засоби аналізу загроз та засоби контролю запобігання загрозам від Check Point.

Ціни на Check Point CloudGuard

Ціни Check Point можуть відрізнятися залежно від моделі ліцензування, обсягу трафіку, регіонів, пропускної здатності, доповнень, ведення журналу, сховища та високої доступності.

Як приклад, Check Point моделює середовище хмарного брандмауера

середнього розміру зі 100 ТБ/місяць перевіреного трафіку за ціною $3 227,50/місяць або $38 750/рік.

Lacework FortiCNAP

Найкраще підходить для: оцінки хмарних технологій у поєднанні з виявленням аномалій та покриттям CNAPP Lacework FortiCNAPP – це платформа Fortinet CNAPP, побудована на основі технології Lacework та інтегрована у Fortinet Security Fabric. Вона є найсильнішою, коли оцінка хмарних технологій має бути близькою до виявлення аномалій, безпеки робочих навантажень, контексту вразливостей, відповідності та покриття Kubernetes/контейнерів.

Його головна відмінність полягає в охопленні CNAPP, орієнтованому на виявлення, а не лише на оцінку стану. Покупцям слід ретельно перевіряти поточну упаковку, оскільки назва продукту та ліцензування змінилися після того, як Lacework перейшов до портфоліо Fortinet.

Функції мереживного вишивки FortiCNAPP:

– CNAPP: Поєднує можливості хмарного захисту для різних хмарних середовищ, ідентифікаторів, робочих навантажень, коду, Kubernetes, IaC та виявлення.

– CSPM: Забезпечує видимість хмарних ресурсів, зменшує ризик неправильної конфігурації та підтримує постійну відповідність вимогам.

– CIEM: Забезпечує постійний доступ до хмарних ідентифікаторів та допомагає знизити ризик, пов'язаний з правами доступу.

– Безпека Kubernetes: моніторить середовища Kubernetes на наявність неправильно налаштованих служб, аномальної поведінки та ризикованої діяльності.

Ціни на мереживо FortiCNAPP

FortiCNAPP від Fortinet використовує модель ціноутворення на основі підписки, яка в основному стягується за кожен захищений віртуальний ЦП або за кожного розробника, зі стартовими пакетами, починаючи приблизно від 25 000 доларів США на рік. Фактична вартість залежить від розміру вашої

інфраструктури, рівня розгортання та терміну дії контракту.

Prisma Cloud від Palo Alto Networks

Найкраще підходить для: великих підприємств CNAPP та програм хмарної безпеки з високим рівнем відповідності.

Prisma Cloud створено для корпоративних команд, яким потрібна оцінка хмарних ресурсів від коду до виконання, а не вузький сканер стану. Його найсильнішою перевагою є широка глибина CNAPP, що охоплює CSPM, CIEM, безагентне сканування робочого навантаження, безпеку IaC, видимість API, безпеку контейнерів/Kubernetes, захист середовища виконання та робочі процеси відповідності.

Це найкраще працює, коли оцінка хмарних технологій є частиною більшої програми безпеки підприємства, особливо в організаціях, які вже стандартизовані в мережах Пало-Альто.

Функції Prisma Cloud:

– CSPM: Оцінює стан хмари та ризик неправильної конфігурації.

– CNAPP: Охоплює ризики, пов'язані з кодом, інфраструктурою, середовищем виконання та життєвим циклом програми.

– CIEM: Оцінює хмарні права та ризики для ідентифікації.

– Сканування IaC: Перевіряє інфраструктуру як код перед розгортанням.

– Безпека контейнерів/Kubernetes: охоплює хмарні робочі навантаження та оркестрацію.

Ціни на Prisma Cloud

Prisma Cloud використовує індивідуальне ціноутворення для підприємств. Точна вартість залежить від розміру розгортання, рівнів функцій та узгоджених знижок, але приклади публічних торгових майданчиків та звіти покупців часто показують розгортання вартістю від кількох тисяч доларів до понад 10 000 доларів США на рік.

Qualys TotalCloud

Найкраще підходить для: команд, які поширюють робочі процеси Qualys щодо вразливостей та ризиків активів на оцінку безпеки хмари.

Qualys TotalCloud – це продукт CNAPP всередині платформи Qualys Enterprise TruRisk. Він є найсильнішим, коли програма безпеки вже використовує Qualys для робочих процесів, пов’язаних з вразливістю, відповідністю вимогам, інвентаризацією активів або ризиками, і хоче, щоб оцінка хмарних технологій використовувалася на основі тієї ж моделі.

Це більше актуально для досвідчених команд безпеки, ніж для покупців, які шукають легкий сканер постави.

Функції Qualys TotalCloud:

– Інвентаризація хмарних активів: Виявляє хмарні ресурси та пов'язує їх із ширшою моделлю ризиків Qualys.

– CSPM: Виявляє прогалини в стані та неправильні конфігурації в хмарних середовищах.

– Управління вразливостями: Розширює робочі процеси управління вразливостями у стилі Qualys на хмарні робочі навантаження.

– CIEM: Оцінює права доступу до хмарної інфраструктури та ризики доступу.

Ціни на Qualys TotalCloud

Qualys не публікує стандартного публічного прайс-листа на TotalCloud. Ціноутворення вимагає індивідуальної корпоративної котирування на основі хмарної інфраструктури, кількості ресурсів, вибраних програм та терміну дії контракту. На AWS Marketplace 12-місячний контракт на «Total Cloud Package 16», що охоплює 16 хостів, коштує 5400,00 доларів США.

Microsoft Defender for Cloud

Найкраще для: команд, що працюють в першу чергу з Azure, яким потрібні нативні хмарні технології, Secure Score та панелі інструментів для забезпечення відповідності нормативним вимогам..

Microsoft Defender for Cloud – це очевидний перший рівень оцінювання для середовищ з інтенсивним використанням Azure. Він забезпечує керування станом системи, аналіз контекстних ризиків, виявлення загроз, дотримання нормативних вимог, аналіз шляхів атак, захист робочих навантажень та

інтеграції з Microsoft.

Він може підтримувати AWS та GCP залежно від конфігурації, але найкраще підходить для команд, орієнтованих на Azure, які хочуть проводити оцінку всередині операцій безпеки Microsoft.

Функції Microsoft Defender для хмари:

– Оцінка безпеки: Надає оцінку стану та пріоритетні рекомендації щодо безпеки.

– Базовий CSPM: Microsoft стверджує, що базовий CSPM є безкоштовним і включає постійні оцінки, рекомендації щодо безпеки, оцінку безпеки та бенчмарк безпеки хмарних технологій Microsoft у Azure, AWS та GCP.

– Defender CSPM: Додає розширені можливості контролю стану, такі як безагентне сканування вразливостей, аналіз шляхів атак, контроль стану безпеки з урахуванням даних, контекстуалізація коду в хмарі та інтелектуальний графік безпеки хмари.

– Захист робочого навантаження: Додає плани Defender для серверів, контейнерів, баз даних, сховищ, API, служб програм, Key Vault, Resource Manager та служб штучного інтелекту залежно від увімкненого плану.

Ціни на Microsoft Defender для хмари

Foundational CSPM безкоштовний. Ціна Defender CSPM залежить від розміру хмари та стягує плату лише за кількість серверів, облікових записів сховища, баз даних та безсерверних ресурсів.

Захист робочого навантаження оцінюється окремо за типом захищеного ресурсу, таким як сервери, контейнери, бази даних, сховища, API, служби штучного інтелекту та служби програм.

Splunk Enterprise Security

Найкраще підходить для: команд, які займаються виявленням проблем і хочуть використовувати сигнали оцінки хмарних даних у робочих процесах SIEM..

Splunk Enterprise Security не є суто інструментом оцінки безпеки хмари.

Він належить до цього списку лише для команд, які спрямовують результати аналізу стану, хмарні журнали, сигнали ідентифікації, дані про вразливості та події виявлення в робочі процеси SIEM.

Його цінність полягає в кореляції, розслідуванні, інженерії виявлення, оповіщенні на основі ризиків, аналітиці впливу та робочому процесі SOC. Він повинен отримувати сигнали оцінки хмари, а не замінювати інструменти, які їх генерують.

Функції Splunk Enterprise Security:

– Кореляція SIEM: Централізує дані безпеки та співвідносить сповіщення, журнали, сигнали ідентифікації, хмарну телеметрію та події виявлення.

– Оповіщення на основі ризику: Splunk стверджує, що оповіщення на основі ризику може зменшити обсяг оповіщень до 90%, що важливо, коли сигнали стану та виявлення хмари починають надсилати до однієї й тієї ж черги SOC.

– Аналітика впливу: Постійно ідентифікує активи та користувачів, збагачує результати контекстом та підтримує пріоритезацію ризиків та ідентифікацію впливу.

– SOAR та UEBA: Enterprise Security Premier включає SOAR та UEBA, тоді як Essentials включає SIEM, розвідку загроз, Detection Studio та Exposure Analytics.

Ціни на Splunk Enterprise Security

Корпоративна безпека продається через такі варіанти придбання, як Essentials та Premier, а остаточне ціноутворення формується через продажі Splunk, а не за таблицею цін самообслуговування.

Для порівняння, на сторінці Splunk, присвяченій корпоративній безпеці, наведено знімок бізнес-цінності IDC, який показує 304% рентабельність інвестицій, на 64% швидше виявлення загроз, 12-місячну окупність та 4,89 млн доларів США щорічної економії витрат на безпеку для уніфікованої платформи Splunk TDIR.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для розробки програмних компонентів системи моніторингу, аналізу та оцінки вразливостей у безсерверних хмарних середовищах було обрано мову програмування Python. Це динамічна, інтерпретована об’єктно-орієнтована мова зі строгою динамічною типізацією, офіційний ресурс якої доступний за адресою Python.org.

Python є високоефективним інструментом для створення хмарних мікросервісів, скриптів автоматизованого сканування та інтеграції з платформами хмарних обчислень (Serverless architectures). Завдяки лаконічності синтаксису та високій швидкості розробки, Python дозволяє оперативно реалізувати складну логіку аналізу безпеки, збору метрик і взаємодії з хмарними API без зайвих накладних витрат на написання допоміжного коду.

Особливості виконання та розширюваність

Однією з особливостей Python є його інтерпретована природа: початковий код спочатку компілюється у внутрішній байт-код, який виконується віртуальною машиною (інтерпретатором). У контексті задач моніторингу вразливостей та обробки логів безпеки в реальному часі критично важливими є продуктивність та ефективність обчислень.

Python легко інтегрується з оптимізованими модулями, написаними на C або C++, що дозволяє задіювати високопродуктивні бібліотеки для виконання ресурсомістких криптографічних операцій, глибокого аналізу пакетів або паралельної обробки великих масивів даних телеметрії безсерверних додатків.

Стандартна бібліотека та екосистема для хмарних середовищ

Вагомою перевагою Python є його розширювана стандартна бібліотека та велика екосистема сторонніх пакетів, які забезпечують:

– зручну роботу з мережевими протоколами, HTTP-запитами та веб-сервісами;

– взаємодію з хмарними інфраструктурами (наприклад, AWS Lambda,

Google Cloud Functions, Azure Functions) через офіційні SDK;

– парсинг форматів даних (JSON, YAML, XML), які використовуються у конфігураціях безсерверних додатків;

– побудову аналітичних моделей для виявлення аномалій та потенційних вразливостей у коді функцій.

Для розробки та налагодження компонентів системи використовуються сучасні інтегровані середовища та редактори коду, такі як Visual Studio Code, PyCharm, а також вбудовані засоби автоматизації тестування й контролю якості коду. Повна крос-платформоспрямованість мови забезпечує безпроблемне розгортання розроблених модулів моніторингу як у локальних середовищах розробників, так і в хмарній інфраструктурі.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на

екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Згідно з останніми дослідженнями ринку, ринок безпеки додатків швидко зростає, оскільки організації збільшують інвестиції в інструменти, що захищають сучасні програмні середовища. Прогнозується, що обсяг ринку зросте з 13,61 млрд доларів США у 2025 році до 14,83 млрд доларів США у 2026 році, досягнувши 28,11 млрд доларів США до 2031 року. Це відповідає сукупному річному темпу зростання (CAGR) на рівні 13,64% між 2026 і 2031 роками.

Декілька тенденцій прискорюють попит на інструменти безпеки додатків:

– Зростання кількості та складності веб-, мобільних та API-атак: Зловмисники часто націлюються на незахищені API, зламані засоби керування авторизацією та надмірне викриття даних. У відповідь організації впроваджують динамічні та інтерактивні інструменти тестування, які імітують атаки на запущені програми.

– Нормативні вимоги: такі стандарти, як PCI-DSS 4.0, GDPR та Закон про цифрову операційну стійкість (DORA), вимагають ретельніше тестування додатків та звітності про відповідність. Наприклад, PCI-DSS 4.0 запровадив нові вимоги, включаючи обов'язковий аналіз складу програмного забезпечення для систем, що обробляють платіжні дані.

– Зростання використання сторонніх API та інтеграцій SaaS: це збільшило ризик для ланцюга поставок. Багато програм залежать від десятків зовнішніх сервісів, що спонукає організації розгортати інструменти, які сканують залежності та контролюють взаємодію API.

Кілька сегментів домінують у витратах на ринку безпеки додатків. На рішення для безпеки припадає 61,48% доходів ринку. Хмарні платформи становлять 57,81% витрат, що відображає зростаюче використання хмарних

середовищ розробки. Що стосується типів тестування, статичне тестування безпеки додатків (SAST) займає найбільшу частку – 36,38% доходів.

Категорії інструментів безпеки програм

Захист веб-застосунків та API (WAAP)

Рішення для захисту веб-застосунків та API (WAAP) розширюють традиційні можливості WAF для захисту не лише веб-застосунків, але й API, які зараз є основними цілями для зловмисників. Інструменти WAAP включають захист від повного спектру веб-загроз та API, включаючи атаки ін'єкцій, порушення автентифікації, витік даних та атаки, специфічні для API, такі як зловживання бізнес-логікою. Перевіряючи та фільтруючи як веб-трафік, так і трафік API, WAAP забезпечує цілісну безпеку на всіх сучасних поверхнях застосунків, часто включаючи запобігання ботам, захист від DDoS-атак та розширену аналітику.

Безпека ШІ

Інструменти безпеки на основі штучного інтелекту допомагають захистити програми, що використовують моделі штучного інтелекту та машинного навчання. Ці інструменти усувають унікальні ризики, такі як швидке впровадження, отруєння даних, атаки з боку зловмисного вхідного сигналу та ненавмисне розкриття конфіденційної інформації через виходи, згенеровані штучним інтелектом. Вони часто надають такі можливості, як очищення вхідних даних, моніторинг поведінки моделі та виявлення дрейфу, щоб забезпечити належну поведінку моделей.

Інтерактивне тестування безпеки застосунків (IAST)

Інтерактивне тестування безпеки додатків (IAST) поєднує аспекти SAST та DAST, моніторячи додатки зсередини під час їх роботи в тестових або контрольно-профілактичних середовищах. Агенти IAST вбудовані в додаток, що дозволяє їм одночасно аналізувати дані виконання, виконання коду та взаємодію з користувачами. Це дозволяє точно ідентифікувати вразливості та їх розташування в коді, зменшуючи кількість хибнопозитивних результатів та добре вписуючись у робочі процеси DevOps.

Динамічне тестування безпеки застосунків (DAST)

Інструменти динамічного тестування безпеки додатків (DAST) оцінюють додатки ззовні під час їх виконання, імітуючи, як зловмисник може їх використати в реальному середовищі. DAST зосереджується на виявленні вразливостей, таких як міжсайтовий скриптинг, SQL-ін'єкції та проблеми автентифікації, через реальні HTTP-запити, і не вимагає доступу до вихідного коду, що робить його придатним для тестування «чорної скриньки».

Статичне тестування безпеки додатків (SAST)

Інструменти статичного тестування безпеки додатків (SAST) аналізують вихідний код, байт-код або бінарні файли додатків на наявність вразливостей безпеки без запуску програми. Працюючи на ранніх етапах розробки, SAST дозволяє командам виявляти та усувати вразливості до розгортання та може бути інтегрований безпосередньо в середовища розробки та конвеєри CI/CD. Однак інструменти SAST можуть давати хибнопозитивні результати та вимагати контекстуальних знань для інтерпретації результатів.

Аналіз складу програмного забезпечення (SCA)

Інструменти аналізу складу програмного забезпечення (SCA) виявляють вразливості та проблеми з дотриманням ліцензійних вимог у компонентах з відкритим кодом та сторонніх компонентах, що використовуються в програмі. Автоматично скануючи дерева залежностей, інструменти SCA попереджають розробників про відомі вразливості в зовнішніх пакетах і рекомендують безпечніші альтернативи або виправлення, а також відстежують ліцензії з відкритим кодом, щоб уникнути юридичних проблем і проблем із дотриманням вимог.

Kubernetes WAF

Брандмауери веб-застосунків (WAF) Kubernetes – це спеціалізовані інструменти, призначені для захисту контейнерних застосунків, що працюють у кластерах Kubernetes. На відміну від традиційних WAF на межі мережі або застосунків, WAF Kubernetes розгортаються безпосередньо в середовищах Kubernetes та інтегруються з робочими навантаженнями кластера, захищаючи

від 10 найкращих загроз OWASP шляхом фільтрації та моніторингу HTTP-трафіку та використання переваг вбудованих функцій Kubernetes, таких як мітки, простори імен та виявлення служб.

3.2 Розробка структурної схеми

Розроблене під час виконання магістерської роботи програмне забезпечення – це хмарна платформа, яка об’єднує кілька модулів захисту в один інтегрований сервіс на базі штучного інтелекту для веб-додатків та API. Модулі обмінюються даними про атаки та реагують разом, а розроблене програмне забезпечення застосовує машинне навчання до них для виявлення аномалій та автоматичного оновлення політик безпеки. Він надається через архітектуру, яка розроблена для забезпечення захисту в локальних, приватних, публічних та гібридних хмарних середовищах, включаючи Kubernetes.

Основні характеристики включають:

– Комплексний захист: поєднує WAF, захист API, керування ботами, захист на стороні клієнта та захист від DDoS-атак рівня 7 в одному сервісі, завдяки чому модулі обмінюються даними про атаки та реагують разом.

– Захист API: Автоматично виявляє API, створює індивідуальні політики безпеки та використовує безперервне відображення на основі штучного інтелекту для виявлення та зупинки атак на бізнес-логіку в режимі реального часу.

– Захист на основі машинного навчання: використовує автоматизовану модель позитивної безпеки та поведінкові алгоритми для виявлення аномалій, блокування атак нульового дня та зниження рівня хибнопозитивних результатів.

– Керування ботами: Фільтрує корисну та шкідливу активність ботів на вебсайтах, у мобільних додатках та API, розрізняючи людський трафік від автоматизованого трафіку.

– Покриття на стороні клієнта та DDoS-атак: додає захист на стороні клієнта від атак скриптів ланцюга поставок та пом'якшення поведінкових наслідків DDoS-атак на рівні додатків на основі HTTP.

– Послідовна керована доставка: застосовує однакові політики в усіх середовищах хостингу через SecurePath без зміни маршруту чи спільного використання SSL-сертифіката та працює як керована служба з командою екстреного реагування.

– Захист LLM та захоплення облікового запису: Включає опцію брандмауера LLM для захисту на рівні оперативного доступу від генеративного використання ШІ та поведінкового виявлення спроб розподіленого захоплення облікового запису.

Обмеження:

– Глибина звітності: Деякі користувачі хотіли б мати більше глибини та можливостей налаштування у вбудованих звітах та інформаційних панелях.

– Крива навчання: Опанування користувацьких правил та налаштування політик може потребувати часу та певних знань у сфері безпеки.

– Початкове налаштування: Адаптація та початкове налаштування можуть вимагати координації та зусиль для правильного виконання.

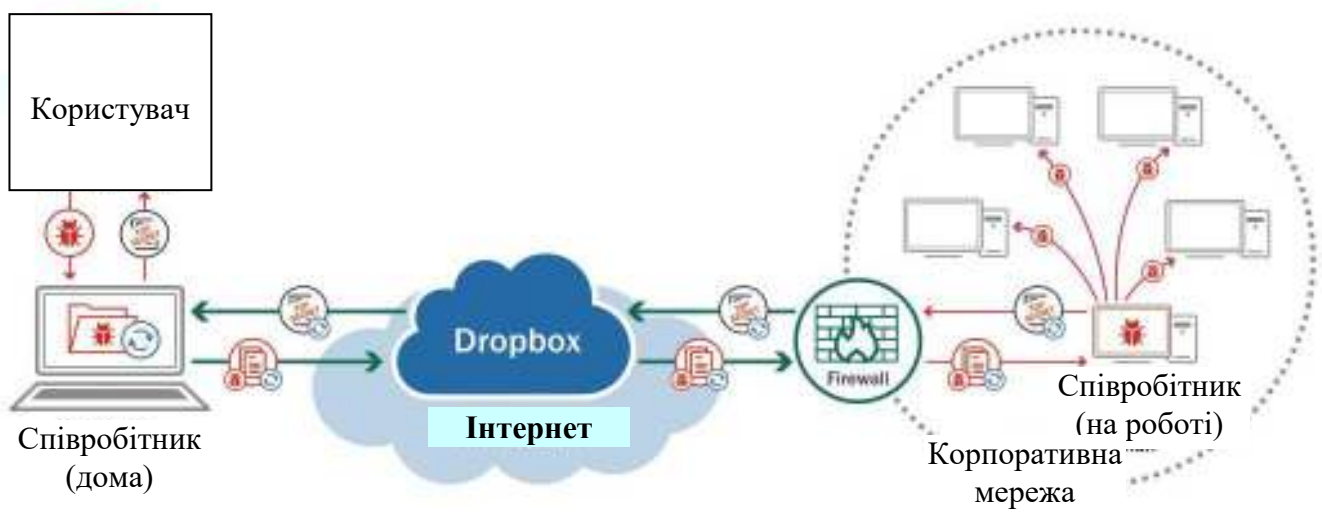


Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

Для реалізації інтелектуальних компонентів системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних середовищах доцільно застосувати апарат штучних нейронних мереж. Розглянемо архітектурні особливості та принцип дії нейрокомп'ютерної мережі зустрічного поширення, яка використовується для аналізу безпеки.

Нейрокомп'ютерна мережа зустрічного поширення у задачах моніторингу безпеки

Нейромережі зустрічного поширення ефективно застосовуються для асоціативного зв'язування вхідних вектором (що містять, наприклад, параметри запитів, логери виконання функцій або системні мітки телеметрії) із цільовими вихідними векторами, які класифікують стан безпеки додатка (безпечний стан, аномалія, потенційна вразливість).

У процесі навчання вхідні та вихідні вектори (двійкові або неперервні) синхронізуються. На етапі експлуатації розгорнутої безсерверної системи подання неповного, зашумленого або спотвореного вхідного вектора (наприклад, фрагментарних логів виклику хмарної функції) дозволяє мережі генерувати коректний вихідний результат завдяки високій узагальнюючій здатності. Це робить такий архітектурний підхід придатним для задач розпізнавання шаблонів атак, відновлення неповних даних телеметрії та виявлення прихованих векторів вразливостей у реальному часі.

Структура та архітектура мережі

На рисунку 3.2 наведено спрощену схему мережі прямої дії зустрічного поширення, яка ілюструє базові функціональні зв'язки між її шарами в контексті аналізу безсерверного середовища:

– Шар 0 (вхідний шар) – виконує функції розподілу та маршрутизації вхідних параметрів моніторингу без виконання обчислювальних перетворень.

– Шар 1 (шар Кохонена) – формує конкурентний шар (кластеризацію вхідних станів). Кожен нейрон нульового шару з'єднаний із нейронами шару

Кохонена через систему вагових коефіцієнтів, що утворюють матрицю ваг W. Цей шар відповідає за виділення ключових закономірностей у потоках подій безпеки.

– Шар 2 (шар Гроссберга) – вихідний шар асоціативного відображення. Нейрони шару Кохонена з'єднані з нейронами шару Гроссберга за допомогою матриці ваг V.

Головна відмінність цієї парадигми від інших типів нейромереж полягає в специфіці операцій, що виконуються нейронами шарів Кохонена та Гроссберга, що забезпечує високу швидкість адаптації системи до динамічних умов виконання хмарних мікросервісів.

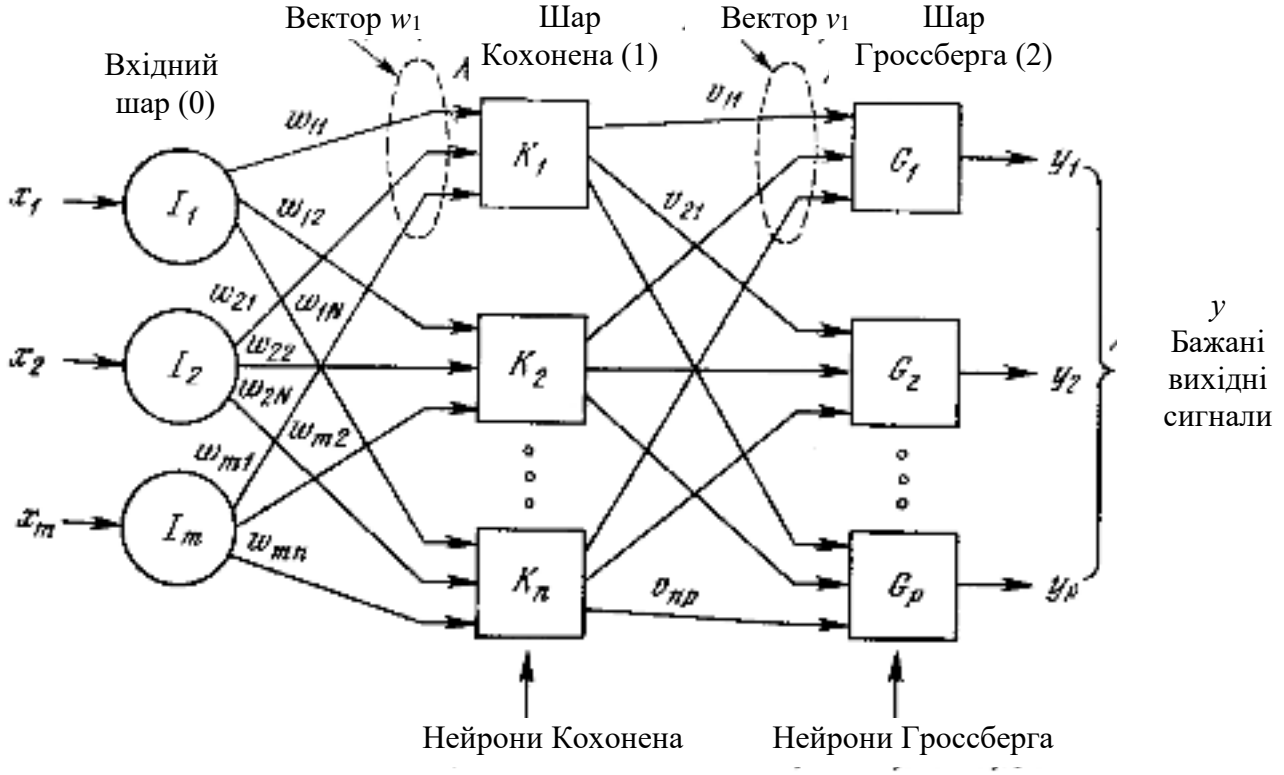


Рисунок 3.2 – Мережа із зустрічним розпізнаванням без зворотних зв'язків

Подібно до більшості штучних нейронних мереж, модель зустрічного поширення функціонує у двох базових режимах:

– режимі нормальної експлуатації – здійснюється прийом вхідного вектора параметрів X (що містить, наприклад, телеметричні дані виконання функцій, системні логи або мітки активності) із наступною генерацією вихідного вектора Y, який формує оцінку стану безпеки та наявності вразливостей;

– режимі навчання – виконується подавання вхідних векторів із наступним адаптивним коригуванням вагових коефіцієнтів для забезпечення необхідної реакції системи на потенційні загрози чи аномалії.

Розробка нейрокомп'ютерной мережі в застосуванні до моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах

Під час розв'язання задачі моніторингу та оцінки вразливостей додатків у безсерверних хмарних середовищах (з урахуванням аналізу захищеності переданих даних) обрано концептуальний підхід, за якого на етапі навчання формується повний набір вхідних даних: відкритий текст, ключ, алгоритм шифрування та відповідний криптотекст (закритий текст). Натомість на етапі безпосереднього моніторингу та оперативної оцінки вразливостей система оперує виключно закритим текстом.

На етапі розпізнавання здійснюється виділення з масивів криптотексту знайомих системі патернів (зразків) із подальшою їхньою ідентифікацією окремими нейронами або нейронними ансамблями на вищих рівнях обробки. Як на етапі навчання, так і під час розпізнавання вхідні вектори мають стохастичний або нечіткий характер – спостерігається певний розкид значень векторів, що належать до одного класу. У зв'язку з цим нейромережева структура повинна володіти властивістю статистичного усереднення. Водночас можлива ситуація, коли вектори розташовані в безпосередній близькості один до одного, але належать до принципово різних класів (або сценаріїв загроз). Це вимагає від нейромережі здатності розрізняти тонкі структурні відмінності між ними. Додатковою ключовою вимогою до нейромережевих модулів низького

рівня обробки сигналів є реалізація навчання без учителя (*unsupervised learning*), що забезпечує здатність системи до самостійної кластеризації та розділення вхідних сигналів.

Для виконання функцій класифікації та поділу вхідного криптотексту на категорії застосовуються різні нейромережеві алгоритми. У теорії штучних нейронних мереж виділяють три базові математичні моделі такого поділу:

1. Поділ вхідних даних гіперплощинами (простий персептрон). Застосування цього підходу доцільне виключно для задач із високим рівнем лінійності. Наприклад, простий персептрон здатен успішно вирішити задачу лінійного розділення точок (0,0) та (1,1) на два окремі класи у двовимірному просторі сигналів. Проте він принципово не здатний вирішити нелінійні задачі, класичним прикладом яких є проблема «виключного АБО» (XOR), де точки (0,0) та (1,1) належать до першого класу, а точки (0,1) та (1,0) – до другого.

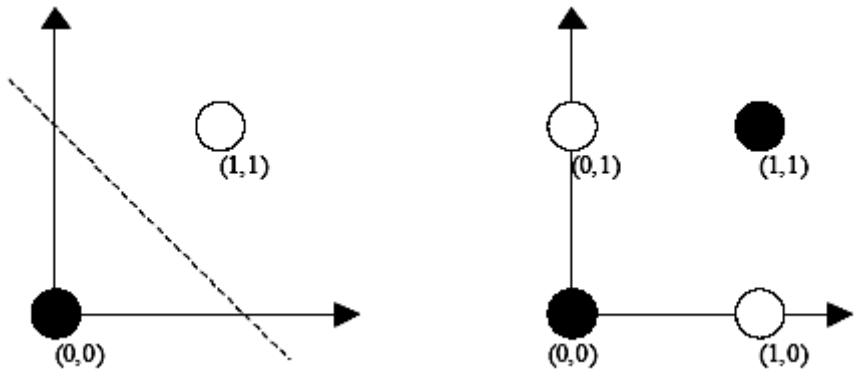


Рисунок 3.3– Теорема Мінського

2. Поділ вхідних даних гіперповерхнями (багатошарові персептрони). Послідовне об'єднання шарів, подібних до простого персептрона, дозволяє комбінувати окремі гіперплощини та формувати складні нелінійні гіперповерхні, зокрема замкнуті області. За умови достатньої кількості нейронів така мережа теоретично здатна здійснювати класифікацію сигналів практично будь-якого рівня складності. Проте практичне застосування багатошарових персептронів обмежується трудомісткістю їхнього навчання. Попри розробку ефективного алгоритму зворотного поширення помилки (*backpropagation*),

процес навчання вимагає значних обчислювальних витрат і не завжди гарантує досягнення глобального мінімуму функції втрат через ризик потрапляння в локальні мінімуми.

3. Пошук найбільшої відповідності (мінімальної кутової або лінійного відхилення). У разі використання нормалізованих вхідних векторів усі вони розміщуються безпосередньо на поверхні багатовимірної гіперсфери. Цим вимогам повною мірою відповідає мережа зустрічного поширення (*Counterpropagation Network*), яка архітектурно є комбінацією двох класичних парадигм – самоорганізованої карти Кохонена та шару Гроссберга. У процесі навчання вхідні вектори (параметри моніторингу безпеки) асоціюються з відповідними вихідними станами. На етапі функціонування розгорнутої системи подання неповних, зашумлених або злегка спотворених вхідних даних дозволяє отримати коректний результат завдяки високій узагальнюючій здатності мережі.

Схематичне зображення архітектури мережі зустрічного поширення наведено на рисунку 3.4.

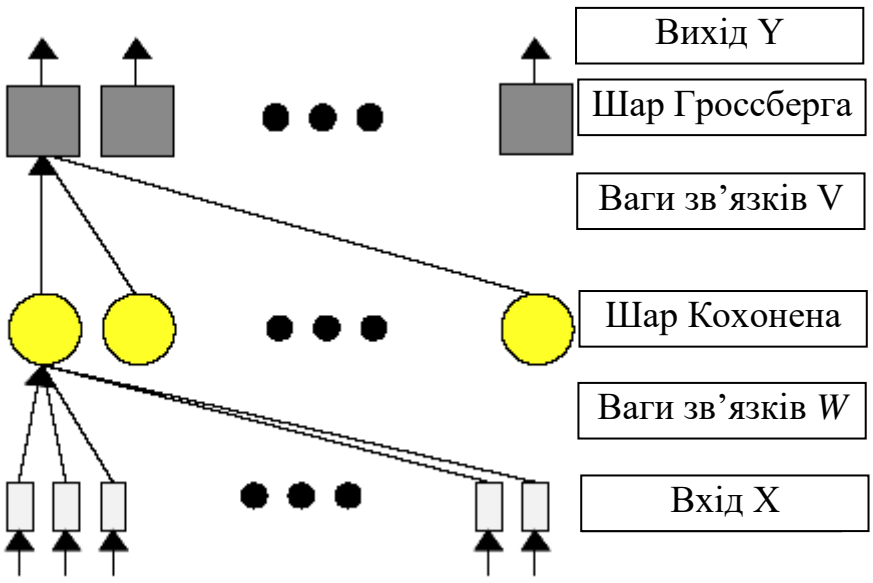


Рисунок 3.4 – Мережа зустрічного поширення

Поширення інформаційних потоків у такій мережі реалізується таким чином: вхідний вектор параметрів попередньо нормується до одиничної

довжини та подається на вхідний шар, звідки розподіляється через матрицю вагових коефіцієнтів W. Кожен нейрон шару Кохонена обчислює сумарне значення на своєму вході та залежно від стану сусідніх елементів набуває активного або неактивного стану (значення 1,0 або 0,0 відповідно).

Нейрони цього шару функціонують на основі конкурентної взаємодії: в результаті ітераційного процесу активним залишається лише один нейрон (або локальна група, так званий «пухирець активності»). Цей класичний механізм латерального гальмування хоч і є ефективним, проте вимагає значних обчислювальних ресурсів. Тому в розробленій моделі цей підхід оптимізовано шляхом прямого пошуку нейрона з максимальним рівнем активності, якому присвоюється одиничне значення (1,0), тоді як решта нейронів переводяться в нульовий стан (0,0). Завдяки цьому спрацьовує саме той нейрон, вектор вагових коефіцієнтів якого виявляється найближчим до вхідного вектора в багатовимірному просторі ознак безпеки.

Алгоритм навчання

Алгоритм навчання нейромережевої моделі:

– Вхідні дані: навчальна вибірка (набір вхідних векторів телеметрії та параметрів безпеки).

– Вихідні дані: скоректовані вагові зв'язки мережі.

1. Подати вхідний вектор на вхідні елементи нейромережі.

2. Виконувати ітераційні кроки до досягнення стабільного стану системи.

3. Здійснити корекцію вагових зв'язків для всіх вузлів мережі відповідно до встановлених математичних залежностей ((2) або (3)).

4. Повторювати кроки 1–3 для кожного вхідного вектора навчальної вибірки.

Розроблена система моніторингу та оцінки вразливостей додатків у безсерверних хмарних середовищах є універсальною, вирізняється низькими вимогами до обчислювальних ресурсів і пам'яті, а також демонструє вищу ефективність порівняно з класичними методами захисту та аналізу безпеки хмарних FaaS-платформ. До ключових переваг системи належать: висока

швидкість аналізу, гнучкість адаптації до динамічних умов виконання та універсальність, оскільки існуючі традиційні підходи часто є вразливими до подібних векторів аналізу.

Аналіз функціональної схеми системи

Функціональну схему розробленої системи наведено на рисунку 3.5. Відповідно до схеми, архітектура комплексу складається з таких базових елементів:

– додатки у хмарних сервісах (Cloud-services);

– нейронна мережа зустрічного поширення;

– блок порівняння вхідних даних із еталонними патернами (образами), відомими системі;

– модуль навчальної вибірки;

– ідентифіковані вразливі додатки у хмарних сервісах.

На початковому етапі на вхід системи подається навчальна вибірка, за допомогою якої здійснюється процедура навчання нейронної мережі. У подальшому під час моніторингу безпеки хмарних додатків виконується аналіз зашифрованих потоків даних та параметрів виконання. Нейронна мережа ідентифікує використовувані алгоритми та оцінює потенційні вразливості за рахунок порівняння поточних вхідних даних із набором навчених і збережених системою образів.

Після завершення аналізу та оцінки безпеки хмарних компонентів отримана нова інформація інтегрується до бази знань системи. У результаті з кожним наступним циклом моніторингу розширюється інформаційна база для виявлення потенційних загроз і вразливостей.

У якості основного ядра системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних середовищах використовується штучна нейронна мережа зустрічного поширення (*Counterpropagation Network*), яка містить три базові рівні:

– вхідний шар (X) – для прийому вхідних параметрів і телеметрії;

– шар Кохонена (K) – для кластеризації та класифікації станів безпеки;

– шар Гроссберга (V) – для формування вихідної оцінки системи.

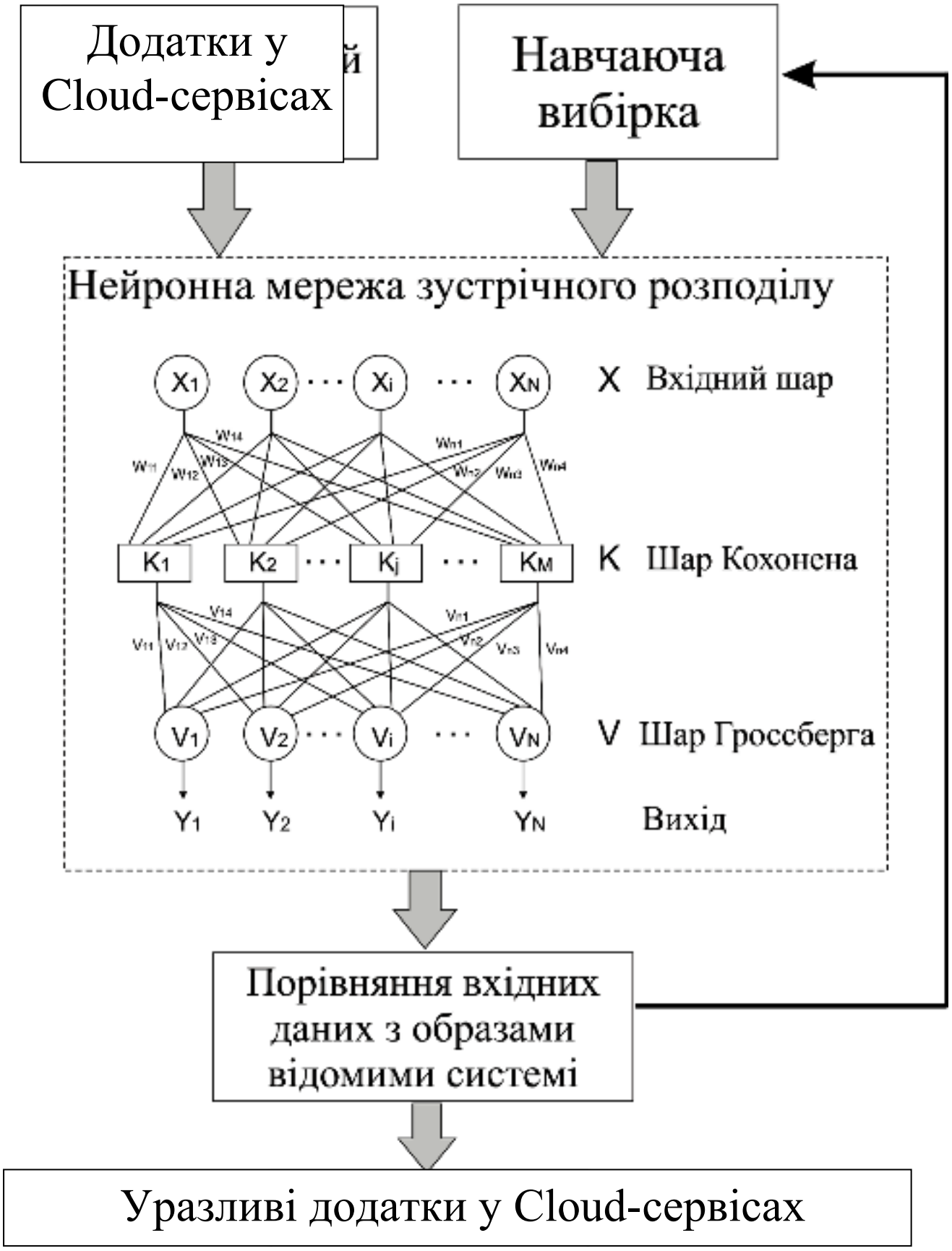


Рисунок 3.5 – Функціональна схема системи

Архітектура зустрічного поширення вдало поєднує можливості інформаційного узагальнення, притаманні карткам Кохонена, із простотою навчання вихідної структури Гроссберга. Цей підхід є оптимальним для швидкого моделювання та розгортання систем на початкових етапах досліджень із можливістю подальшого переходу (за необхідності) на складніші ресурсомісткі методи на кшталт зворотного поширення помилки.

Нейронна мережа зустрічного поширення навчається на основі вибірки пар векторів (X,Y), вирішуючи задачу апроксимації відображення XY. Важливою перевагою цієї мережі є здатність до автоасоціативного відображення сукупності XY у себе. Завдяки високій узагальнюючій здатності це дозволяє відновлювати повну пару (XY) за наявності лише одного відомого компонента (X або Y). Так, під час розпізнавання за поданням виключно вхідного вектора X (при нульовому початковому стані Y) здійснюється пряме відновлення цільового результату, і навпаки. Здатність розв'язувати як прямі, так і зворотні задачі, а також гібридні сценарії відновлення відсутніх компонентів телеметрії перетворює цю архітектуру на унікальний інструмент аналізу безпеки хмарних середовищ.

Режими функціонування та навчання

У режимі експлуатації (розпізнавання) нейрони шару Кохонена функціонують за принципом конкурентного відбору «переможець забирає все», визначаючи кластер, якому відповідає поточний вхідний образ стану системи. Надалі вихідна зірка шару Гроссберга за сигналом нейрона-переможця формує результуючу оцінку вразливостей.

Навчання вагових коефіцієнтів шару Кохонена реалізується без учителя на основі механізмів самоорганізації. Перед початком процесу вхідний вектор нормується зі збереженням початкового напрямку. Після виконання ітерації визначається активний нейрон-переможець, стан якого встановлюється на рівні одиниці, що дозволяє модифікувати ваги відповідної зірки Гроссберга. Швидкості навчання шарів Кохонена та Гроссберга узгоджуються між собою; при цьому в шарі Кохонена коригуються ваги нейронів у межах локальної

околиці переможця, яка поступово звужується до єдиного елемента.

Навчена нейромережа може функціонувати в режимі інтерполяції, коли в шарі Кохонена активується не один, а кілька нейронів. У цьому випадку рівні їхньої активності нормалізуються пропорційно так, щоб у сумі становити одиницю, а результуючий вихідний вектор визначається як зважена сума виходів усіх активних зірок Гроссберга. Це забезпечує лінійну інтерполяцію між значеннями вихідних векторів, що відповідають суміжним кластерам станів безпеки.

Після детального розгляду всіх функціональних блоків архітектури перейдемо до аналізу діаграми взаємодії процесів, що реалізуються в розробленій системі.

3.4 Розробка діаграми процесів

Для візуалізації процесів обробки даних у межах структурного проєктування використовується діаграма взаємодії процесів. Традиційно на початковому етапі розробки створюється контекстна діаграма потоків даних, яка ілюструє загальну взаємодію системи із зовнішнім середовищем. Надалі ця модель деталізується шляхом поетапної декомпозиції процесів і інформаційних потоків, що дозволяє повноцінно відобразити архітектуру та логіку роботи розроблюваного програмного рішення.

Діаграму процесів розробленої системи наведено на рисунку 3.6. Її детальний аналіз дозволяє простежити характер взаємодії внутрішніх компонентів та маршрутизацію даних усередині системи. У межах проєктування застосовується модель візуалізації потоків даних (Data Flow Diagram, DFD), яка забезпечує наочне графічне представлення обміну інформацією в інформаційному середовищі моніторингу безпеки.



Рисунок 3.6 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Алгоритмічне забезпечення системи моніторингу та оцінювання вразливостей застосунків у безсерверних хмарних обчислювальних середовищах призначене для автоматизованого виявлення потенційних порушень інформаційної безпеки, аналізу захищеності програмних компонентів та визначення пріоритетності усунення виявлених недоліків. Особливістю безсерверної архітектури є виконання прикладного коду в середовищі, інфраструктурою якого керує хмарний провайдер. Розробник застосунку відповідає за програмну логіку, залежності, конфігурацію функцій, дозволи доступу та безпеку оброблення даних. Тому алгоритми моніторингу повинні враховувати не лише можливі вразливості вихідного коду, а й параметри хмарного середовища, через які здійснюється виклик функцій та їхня взаємодія з іншими сервісами.

Загальний порядок функціонування програмного комплексу відображено на рисунку 4.1. Основний алгоритм передбачає отримання інформації про безсерверні застосунки, визначення їхніх компонентів, перевірку програмних залежностей, аналіз конфігурацій безпеки та зіставлення отриманих відомостей із доступними джерелами інформації про вразливості.

Після завершення перевірок виконується оцінювання критичності виявлених проблем. Результати зберігаються у структурованому вигляді та використовуються для формування сповіщень і рекомендацій щодо усунення порушень.

Повторне виконання алгоритму дозволяє відстежувати зміни захищеності застосунків протягом їхнього життєвого циклу.

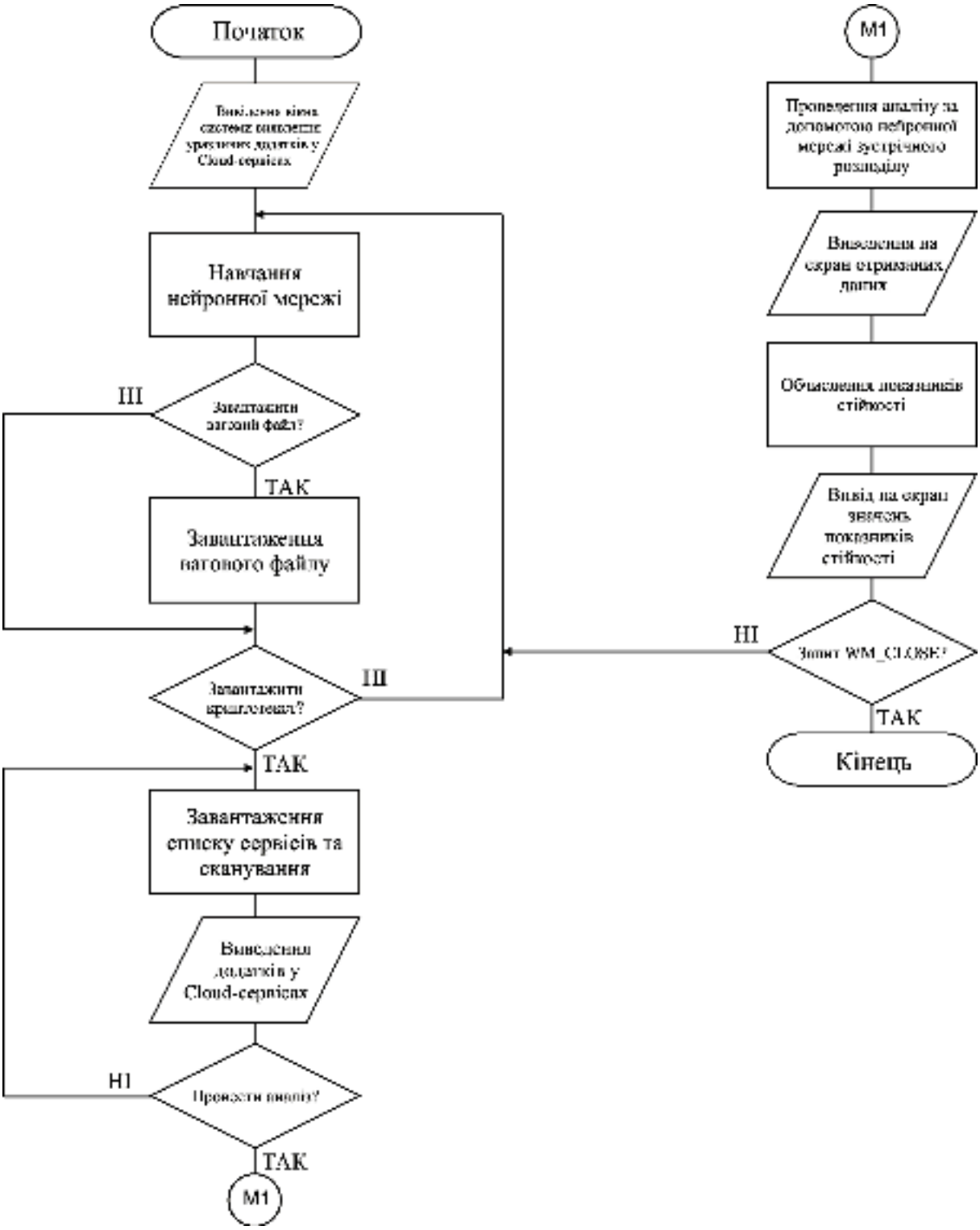


Рисунок 4.1 – Блок-схема основної програми

Для організації моніторингу доцільно використовувати модульну архітектуру, яка розділяє отримання відомостей про хмарні ресурси, аналіз програмного забезпечення та оцінювання ризиків. Компонент інвентаризації

забезпечує виявлення безсерверних функцій і визначення їхніх параметрів. Модуль аналізу залежностей виявляє використані бібліотеки та їхні версії. Компонент перевірки конфігурації контролює налаштування доступу, мережевої взаємодії та захисту секретів. Аналітичний модуль узагальнює отримані результати й визначає пріоритетність виявлених проблем. Такий розподіл функцій дозволяє незалежно розширювати перелік перевірок та адаптувати систему до різних хмарних платформ.

Окремим елементом алгоритмічного забезпечення є підпрограма комплексного оцінювання вразливостей безсерверного застосунку, принцип роботи якої представлено на рисунку 4.2.

Вона отримує результати перевірки програмних компонентів і конфігурацій, визначає наявність підтверджених або потенційних порушень та оцінює їхню значущість для конкретного середовища. Під час прийняття рішення враховуються характер вразливості, доступність відповідного компонента для зовнішніх запитів, надані функції повноваження та можливі наслідки порушення безпеки. Результатом роботи підпрограми є впорядкований перелік виявлених проблем із зазначенням їхнього стану та рекомендованих напрямів усунення.

У безсерверних середовищах необхідно розрізняти вразливості прикладного коду, вразливості сторонніх залежностей і небезпечні конфігурації хмарних ресурсів. Наприклад, використання бібліотеки з відомою вразливістю є проблемою програмної залежності. Надання функції надмірних повноважень є порушенням конфігурації доступу. Недостатня перевірка вхідних даних може створювати вразливість безпосередньо в логіці застосунку. Ці категорії потребують різних способів виявлення та усунення. Для їхнього аналізу може використовуватися класифікація загроз, представлена в матеріалах OWASP Serverless Top 10.

Алгоритм виявлення та інвентаризації безсерверних функцій

Початковим етапом роботи системи є отримання актуального переліку безсерверних компонентів, розгорнутих у контрольованому хмарному середовищі.

Така інвентаризація необхідна для визначення об'єктів перевірки та забезпечення повноти подальшого аналізу. У безсерверній архітектурі кількість функцій може змінюватися внаслідок оновлення застосунків, розгортання нових версій або видалення застарілих компонентів. Тому використання лише статичного переліку ресурсів є недостатнім.

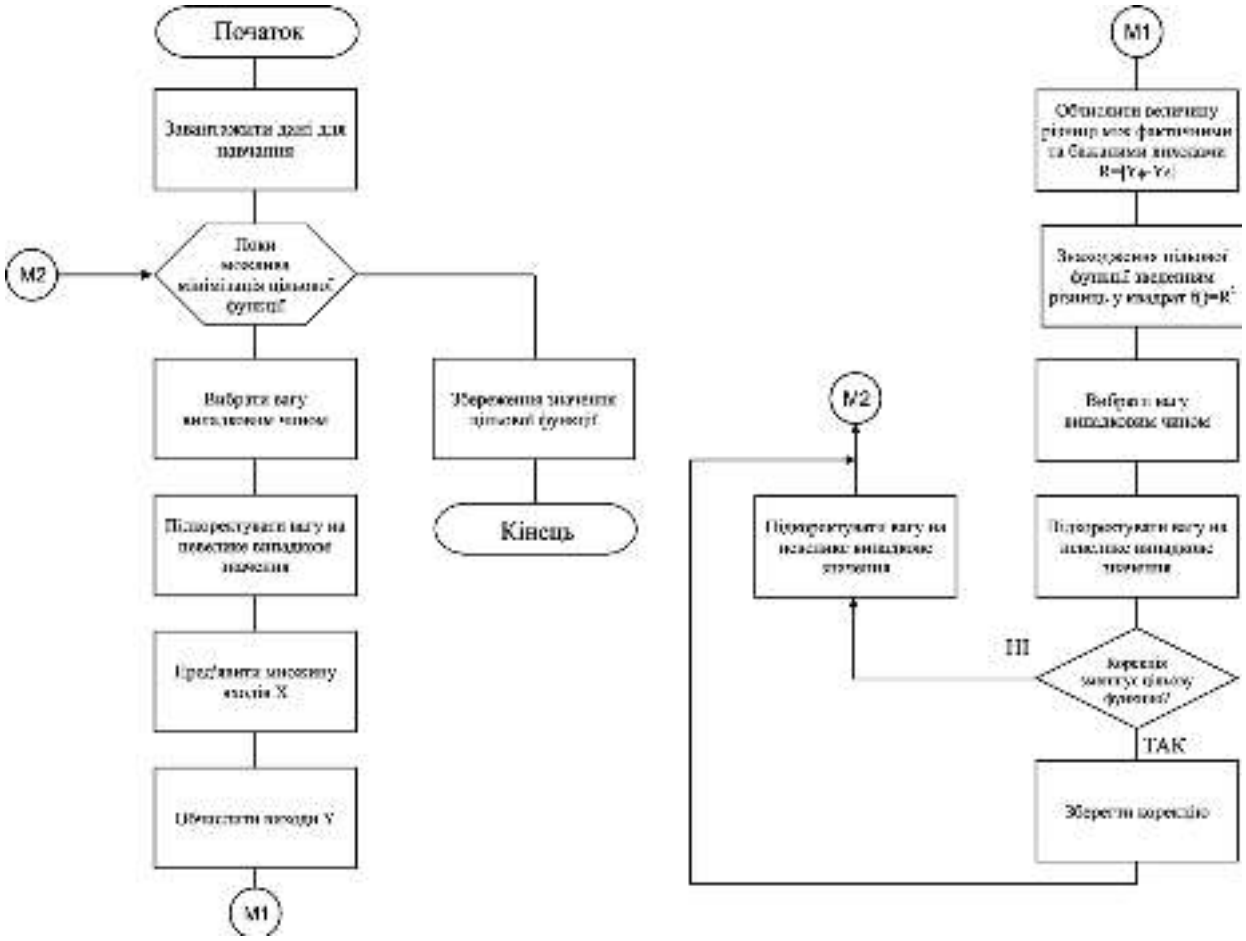


Рисунок 4.2 – Блок-схема роботи підпрограми

Вхідними даними алгоритму є параметри підключення до хмарного провайдера, доступні облікові записи або проєкти та дозволені області розміщення ресурсів. Для взаємодії можуть використовуватися програмні інтерфейси хмарних платформ.

Як можливі середовища розглядаються AWS Lambda, Azure Functions та Google Cloud Run functions. Конкретний адаптер взаємодії повинен враховувати особливості вибраного провайдера та доступних API.

Перед початком інвентаризації перевіряється право керувального компонента отримувати відомості про хмарні ресурси. Для виконання моніторингу доцільно використовувати окрему службову ідентичність з обмеженими повноваженнями. Вона повинна мати доступ лише до необхідних операцій перегляду конфігурації. Надання адміністративних прав для звичайної інвентаризації є небажаним.

Після підтвердження доступу виконується отримання переліку функцій. Для кожного компонента визначаються його ідентифікатор, назва, середовище виконання, регіон розміщення та поточна конфігурація. Додатково можуть отримуватися відомості про версії, псевдоніми, джерела подій, налаштовані дозволи та спосіб пакування програмного коду.

Особливу увагу необхідно приділити розмежуванню логічної функції та її розгорнутих версій. Одна функція може мати декілька версій із різними залежностями та параметрами безпеки. Тому результати перевірки повинні бути пов'язані з конкретною версією або ідентифікатором програмного артефакту.

Отримані відомості нормалізуються та передаються до реєстру контрольованих ресурсів. Якщо функція вже зареєстрована, оновлюються її характеристики. Для нового компонента створюється окремий запис. Відсутність функції під час чергового опитування не повинна автоматично призводити до остаточного видалення її історії. Така ситуація може бути наслідком тимчасової помилки API або зміни прав доступу.

Результатом алгоритму є актуальний перелік безсерверних функцій із зазначенням їхніх характеристик і стану доступності для моніторингу.

Алгоритм отримання та нормалізації конфігураційних даних

Після визначення переліку функцій необхідно отримати інформацію про параметри їхнього виконання. У різних хмарних середовищах структура

конфігурації відрізняється, тому для спільного аналізу потрібне уніфіковане представлення.

Вхідними даними алгоритму є конфігурації функцій, отримані через API відповідного провайдера. До важливих характеристик належать середовище виконання, обсяг виділеної пам'яті, максимальна тривалість виконання, спосіб підключення до мережі, налаштовані тригери та повноваження службової ідентичності.

На першому етапі виконується перевірка коректності отриманої відповіді. Контролюються наявність ідентифікатора ресурсу, підтримуваний тип функції та повнота обов'язкових полів. Якщо API повертає неповну інформацію, відповідні характеристики позначаються як невідомі. Відсутність відомостей не повинна автоматично трактуватися як відсутність небезпечної конфігурації.

Наступним етапом є перетворення даних до внутрішньої моделі. Наприклад, різні провайдери використовують власні способи опису службових ролей, політик доступу та мережевих налаштувань. Для алгоритмів оцінювання необхідно сформувати узгоджені характеристики, які відображають функціональне значення цих параметрів.

Для кожної функції доцільно визначати спосіб виклику. Вона може запускатися за HTTP-запитом, повідомленням із черги, подією об'єктного сховища або за встановленим розкладом. Тип тригера впливає на потенційну доступність функції для стороннього користувача та характер вхідних даних.

Окремо фіксується час отримання конфігурації. У разі зміни параметрів під час сканування необхідно визначити, до якої версії стану належать результати перевірки. Для цього можуть використовуватися часові позначки, версії конфігурації або інші підтримувані провайдером ідентифікатори.

Результатом алгоритму є нормалізовані описи функцій, придатні для застосування єдиних правил перевірки безпеки.

Алгоритм аналізу програмних залежностей

Одним із поширених джерел вразливостей безсерверних застосунків є використання сторонніх бібліотек. Програмна функція може містити невеликий

обсяг власного коду, але використовувати значну кількість залежностей. Частина з них підключається безпосередньо, а інші встановлюються транзитивно.

Для аналізу залежностей необхідно отримати відомості про склад програмного артефакту. Джерелом інформації можуть бути файли опису пакетів, файли фіксації версій, склад розгорнутого пакета або програмний образ. Якщо система має доступ до вихідного репозиторію, аналіз може виконуватися до розгортання функції.

На початковому етапі визначається екосистема програмних залежностей. Для застосунків Python можуть аналізуватися відповідні метадані пакетів та файли залежностей. Для Node.js використовуються дані пакетного менеджера та файли фіксації версій. Аналогічний підхід застосовується для інших підтримуваних середовищ виконання.

Після отримання переліку бібліотек виконується визначення їхніх точних версій. Зазначення лише назви пакета є недостатнім для встановлення наявності відомої вразливості. Особливо важливо враховувати транзитивні залежності, які можуть відрізнятися від безпосередньо оголошених у проєкті.

Для уніфікованого представлення компонентів доцільно формувати Software Bill of Materials (SBOM). Такий документ містить відомості про програмні пакети, їхні версії та зв'язки залежності. Як можливий формат розглядається CycloneDX, який підтримує машинне представлення складу програмного забезпечення.

У разі використання контейнерного способу розгортання аналіз повинен охоплювати не лише прикладні бібліотеки, а й доступні компоненти базового образу. Для функцій, що використовують спільні шари залежностей, необхідно враховувати склад відповідних шарів. Вразливий пакет може бути відсутнім у безпосередньому артефакті функції, але входити до підключеного шару.

Після завершення інвентаризації формується перелік компонентів, придатних для зіставлення з базами відомих вразливостей. Якщо точну версію залежності встановити неможливо, результат позначається як неповний. Такий

випадок потребує додаткової перевірки, а не безумовного висновку про безпечність пакета.

Алгоритм виявлення відомих вразливостей

На основі отриманого складу програмних залежностей виконується пошук опублікованих відомостей про вразливості. Метою алгоритму є встановлення відповідності між конкретними версіями програмних компонентів та відомими проблемами безпеки.

Вхідними даними є перелік пакетів, їхні версії, назви екосистем і додаткові ідентифікатори. Для пошуку можуть використовуватися відкриті бази вразливостей та їхні програмні інтерфейси. Одним із можливих джерел є Open Source Vulnerabilities (OSV), який надає можливість пошуку вразливостей за назвою пакета, екосистемою та версією через OSV API.

Перед формуванням запиту назви компонентів нормалізуються відповідно до правил певної екосистеми. Для зменшення помилок зіставлення доцільно використовувати стандартизовані ідентифікатори пакетів, зокрема Package URL. При цьому необхідно зберігати первинні відомості про компонент для можливості подальшої перевірки.

Після отримання відповіді алгоритм визначає, чи входить встановлена версія пакета до діапазону уражених версій. Для цього використовуються правила порівняння версій відповідної екосистеми. Звичайне лексикографічне порівняння рядків не забезпечує правильного результату для всіх систем версіонування.

Якщо встановлена версія відповідає опису вразливості, створюється запис про виявлену проблему. До нього включаються ідентифікатор вразливості, назва пакета, уражена версія, джерело відомостей та доступні рекомендації щодо оновлення.

Одна вразливість може бути представлена декількома ідентифікаторами. Тому алгоритм повинен враховувати відомі зв'язки між записами, щоб уникати дублювання результатів. Наприклад, один запис OSV може містити посилання на відповідний CVE.

Необхідно також враховувати відсутність повної інформації про виправлені версії. Для деяких вразливостей оновлення може бути недоступним. У такому випадку система не повинна формувати необґрунтовану рекомендацію щодо встановлення довільної новішої версії.

Важливим обмеженням є те, що наявність пакета з відомою вразливістю не завжди означає можливість її експлуатації в конкретній функції. Для цього можуть знадобитися певні конфігурації або виклики вразливого програмного інтерфейсу. Тому результат зіставлення розглядається як встановлення потенційної застосовності вразливості, яка за потреби уточнюється додатковим аналізом.

Результатом алгоритму є перелік відомих вразливостей програмних компонентів із зазначенням ступеня достовірності їхнього зіставлення з фактично розгорнутими залежностями.

Алгоритм статичного аналізу прикладного коду

Відомі вразливості бібліотек становлять лише частину можливих порушень безпеки. Власний код безсерверної функції також може містити недоліки, які виникають через неправильну обробку вхідних даних, використання небезпечних програмних конструкцій або помилки реалізації прикладної логіки.

Для виявлення таких проблем може застосовуватися Static Application Security Testing (SAST). Алгоритм передбачає аналіз вихідного коду без необхідності його виконання в робочому середовищі.

Вхідними даними є вихідні файли застосунку, відомості про мову програмування та набір правил статичного аналізу. На першому етапі визначаються підтримувані типи файлів. Далі виконується синтаксичний аналіз і пошук програмних конструкцій, що відповідають установленим ознакам потенційних вразливостей.

Для безсерверних застосунків особливе значення має аналіз обробників подій. Функція може отримувати дані з HTTP-запиту, повідомлення черги або об'єктного сховища. Якщо отримане значення використовується без належної

перевірки, можуть виникати небезпечні операції з файлами, командами операційної системи або запитами до бази даних.

Більш складний статичний аналіз може використовувати відстеження потоків даних. У такому випадку визначаються джерела потенційно недовірених значень та операції, до яких вони можуть надходити. Якщо між джерелом і небезпечною операцією відсутня необхідна перевірка або обробка, формується повідомлення про можливу вразливість.

Окремо можуть перевірятися небезпечні способи роботи з криптографічними функціями, некоректна перевірка сертифікатів і використання жорстко заданих облікових даних. Для кожної категорії потрібні відповідні правила аналізу.

Статичне виявлення не завжди підтверджує реальну експлуатованість проблеми. Частина результатів може бути хибнопозитивною через особливості виконання програми. Тому система повинна зберігати місце виявлення, опис відповідного правила та необхідний контекст для подальшої перевірки.

Якщо вихідний код недоступний, алгоритм не повинен імітувати виконання SAST за відомостями лише про назву функції. У такому випадку відповідна перевірка позначається як невиконана.

Алгоритм аналізу конфігурації безсерверного середовища

Значна частина ризиків безсерверних застосунків пов'язана з параметрами розгортання. Навіть за відсутності відомих вразливостей у коді неправильна конфігурація може відкривати доступ до захищених ресурсів або дозволяти виконання небажаних операцій.

Вхідними даними алгоритму є нормалізовані характеристики функцій та встановлені правила безпечного конфігурування. На початковому етапі визначаються параметри, які можна перевірити для конкретної хмарної платформи.

До основних об'єктів контролю належать налаштування доступу до функції, параметри автентифікації, дозволені джерела виклику, мережеві обмеження та спосіб зберігання секретів. Окремо перевіряються характеристики

середовища виконання і використання підтримуваних провайдером механізмів захисту.

Для кожного правила визначається очікуваний стан. Наприклад, функція, призначена лише для внутрішнього використання, не повинна мати необмежений публічний доступ. Якщо її конфігурація дозволяє виклик невизначеними зовнішніми суб'єктами, формується відповідне повідомлення.

Під час аналізу необхідно враховувати функціональне призначення компонента. Публічний HTTP-обробник може бути необхідною частиною архітектури. Тому сама наявність зовнішньої адреси не завжди є порушенням. Важливими є правила автентифікації, авторизації та обмеження допустимих операцій.

Результат перевірки повинен містити ідентифікатор функції, назву порушеного правила та параметр конфігурації, який спричинив спрацювання. Якщо властивість відсутня у відповіді хмарного API, перевірка отримує стан недостатності даних.

Таким чином, алгоритм дозволяє виявляти небезпечні конфігурації незалежно від наявності доступу до вихідного коду застосунку.

Алгоритм перевірки повноважень безсерверних функцій

Безсерверні функції часто взаємодіють із базами даних, об'єктними сховищами, чергами повідомлень та іншими хмарними сервісами. Для виконання відповідних операцій їм надаються певні повноваження. Надмірні права можуть збільшувати наслідки експлуатації вразливості прикладного коду.

Алгоритм аналізу повноважень починається з визначення службової ідентичності, від імені якої виконується функція. Для AWS Lambda це може бути execution role, для інших платформ використовуються відповідні механізми керування ідентичностями.

Після отримання відомостей про ідентичність виконується аналіз призначених політик. Визначаються дозволені дії, ресурси та умови застосування прав. Особлива увага приділяється дозволам, які поширюються на всі ресурси

певного типу або передбачають необмежене виконання адміністративних операцій.

Для правильної оцінки необхідно враховувати не лише безпосередньо призначені політики, а й інші механізми, які впливають на ефективні повноваження. У різних хмарних середовищах це можуть бути ресурсні політики, організаційні обмеження та додаткові умови доступу.

Завдання алгоритму полягає у визначенні потенційно надмірних прав. Наприклад, функція, яка повинна лише читати конкретний об'єкт зі сховища, не потребує повноважень на видалення всіх об'єктів у хмарному обліковому записі.

Для точнішого оцінювання права можуть зіставлятися з відомими функціональними залежностями. Якщо сервіс взаємодіє лише з визначеною чергою, наявність широких дозволів на керування іншими ресурсами розглядається як потенційне порушення принципу мінімальних привілеїв.

Проте відсутність спостережень про використання певного дозволу не доводить його непотрібність. Функція може виконувати рідкісні адміністративні або аварійні операції. Тому остаточне скорочення повноважень потребує перевірки відповідності функціональним вимогам.

Результатом алгоритму є перелік потенційно надмірних повноважень із зазначенням ресурсів, на які вони поширюються, та рекомендованих напрямів обмеження.

Алгоритм аналізу джерел подій і зовнішньої доступності

У безсерверній архітектурі виконання функцій ініціюється подіями. Джерелом може бути HTTP-запит, повідомлення брокера, зміна об'єкта у сховищі або внутрішня подія хмарної платформи. Характер джерела визначає можливі способи надходження недовірених даних.

Для аналізу використовується інформація про тригери, пов'язані ресурси та правила виклику функцій. На першому етапі визначаються всі доступні джерела запуску. Далі виконується класифікація залежно від можливості ініціювання події зовнішнім суб'єктом.

Для HTTP-функцій перевіряється спосіб автентифікації та наявність обмежень на виконання запитів. Якщо функція доступна без автентифікації, необхідно визначити, чи відповідає це її призначенню.

Для функцій, які обробляють повідомлення з черг, аналізуються дозволи на передавання повідомлень до відповідного ресурсу. Якщо сторонній суб'єкт може створювати повідомлення, він потенційно здатний впливати на вхідні дані функції.

Аналогічний підхід застосовується до подій об'єктних сховищ. Наявність внутрішнього тригера не гарантує, що вміст події сформований довіреним джерелом. Якщо користувач може завантажувати файли до сховища, обробник цих файлів повинен розглядати їхній вміст як потенційно недовірений.

Для комплексного аналізу доцільно формувати граф зв'язків між функціями, джерелами подій та ресурсами даних. Такий граф дозволяє визначати можливі шляхи поширення впливу від зовнішньої точки входу до внутрішніх сервісів.

Результатом алгоритму є характеристика доступності функції та перелік потенційних напрямів надходження небезпечних даних.

Алгоритм перевірки використання секретів

Для взаємодії з іншими сервісами безсерверні функції можуть використовувати паролі, ключі API, токени доступу та інші службові секрети. Неправильне зберігання таких відомостей створює ризик їхнього розкриття.

Алгоритм перевірки починається з аналізу способів передавання конфігураційних параметрів функції. Вхідними даними є доступні метадані середовища виконання, опис програмного артефакту та результати статичного аналізу коду.

Якщо функція містить жорстко задані облікові дані, алгоритм формує повідомлення про потенційну вразливість. Для виявлення можуть використовуватися шаблони характерних ключів, аналіз структури значень та перевірка їхнього контексту.

Необхідно враховувати, що не кожний рядок, який відповідає формату секрету, є чинним обліковим параметром. Тому результат повинен містити ознаки, необхідні для додаткового підтвердження, без відображення повного значення секрету.

Окремо перевіряється використання спеціалізованих механізмів керування секретами. За наявності відповідної інтеграції функція може отримувати секрет під час виконання через захищений програмний інтерфейс.

При цьому сама наявність менеджера секретів не гарантує правильність захисту. Необхідно також аналізувати права доступу до секретів та можливість їхнього використання іншими функціями.

Виявлені значення секретів не повинні зберігатися у звичайному журналі системи моніторингу. Для підтвердження результату допускається використання обмежених метаданих або захищеного ідентифікатора знахідки.

Результатом алгоритму є перелік потенційних порушень керування секретами з відповідними рекомендаціями щодо їхнього усунення.

Алгоритм оцінювання критичності та пріоритетності вразливостей

Після завершення перевірок формується сукупність виявлених проблем безпеки. Для практичного використання результатів необхідно визначити порядок їхнього усунення. Просте сортування за кількістю вразливостей не дозволяє врахувати реальну небезпеку для конкретного застосунку.

Вхідними даними алгоритму є характеристики знайденої проблеми, відомості про відповідний ресурс та результати аналізу доступності функції. Для відомих вразливостей програмних компонентів можуть використовуватися оцінки Common Vulnerability Scoring System (CVSS).

CVSS характеризує технічну критичність вразливості за визначеною методикою. Водночас базова оцінка не відображає всіх особливостей конкретної інфраструктури. Тому для визначення пріоритетності необхідно враховувати додаткові характеристики середовища, зокрема доступність компонента, його призначення та значущість оброблюваних даних. Принципи оцінювання наведено у специфікації CVSS.

Для уточнення пріоритетності може використовуватися Exploit Prediction Scoring System (EPSS). Цей показник характеризує прогнозовану ймовірність спостереження експлуатації відомої вразливості протягом наступних тридцяти днів. EPSS не є самостійною оцінкою наслідків порушення або повного ризику конкретної системи.

Алгоритм оцінювання спочатку визначає наявність підтверджених характеристик вразливості. Якщо доступна відповідна оцінка CVSS, вона використовується як один із показників. Для вразливостей без стандартизованої оцінки застосовується окрема категоризація відповідно до характеру порушення.

Далі аналізується доступність компонента. Наприклад, вразливість функції, що обробляє зовнішні HTTP-запити, може потребувати вищого пріоритету, ніж аналогічна проблема у внутрішньому компоненті з обмеженим доступом. Проте конкретне рішення залежить від умов експлуатації.

Наступним етапом є оцінювання можливих наслідків. Якщо функція має право змінювати важливі інформаційні ресурси або працює з конфіденційними даними, потенційний вплив порушення може бути значнішим.

Для прийняття рішення доцільно використовувати багатокритеріальну модель пріоритетності. Вона може враховувати технічну критичність, імовірність експлуатації, зовнішню доступність, обсяг повноважень і значущість ресурсу. Конкретні вагові коефіцієнти повинні визначатися за результатами перевірки моделі, а не встановлюватися довільно.

Для проблем, які вже мають підтверджені ознаки активної експлуатації, доцільно передбачити підвищений пріоритет. При цьому необхідно враховувати достовірність джерела відповідної інформації.

Результатом алгоритму є впорядкований перелік проблем безпеки з установленими рівнями пріоритетності та поясненням основних чинників прийнятого рішення.

Алгоритм кореляції та усунення дублювання результатів

Під час комплексного моніторингу одна проблема може бути виявлена декількома механізмами. Наприклад, відома вразливість бібліотеки може

одночасно з'явитися в результатах перевірки залежностей, аналізу програмного артефакту та зовнішнього сканування.

Для запобігання дублюванню необхідно передбачити алгоритм кореляції. Він отримує результати різних перевірок і визначає, чи відповідають вони одній проблемі в одному програмному ресурсі.

На першому етапі зіставляються ідентифікатори функцій і версій. Далі перевіряються відомості про вразливість, програмний компонент та місце виявлення. Для відомих вразливостей можуть використовуватися CVE або інші стандартизовані ідентифікатори.

Якщо декілька записів описують ту саму проблему в одному артефакті, вони об'єднуються в одну логічну знахідку. При цьому зберігаються відомості про всі джерела виявлення.

Необхідно відрізняти повторне виявлення тієї самої проблеми від її наявності в декількох незалежних функціях. Якщо одна вразлива бібліотека використовується різними застосунками, це можуть бути окремі об'єкти усунення з різними власниками та пріоритетами.

Для кожної знахідки формується стабільний внутрішній ідентифікатор. Він дозволяє відстежувати її стан між повторними скануваннями, навіть якщо змінюється порядок отримання результатів.

Результатом алгоритму є узгоджений перелік проблем безпеки, придатний для подальшого моніторингу та керування їхнім усуненням.

Алгоритм безперервного моніторингу змін захищеності

У безсерверному середовищі конфігурація функцій може змінюватися без зупинки всього застосунку. Нові версії коду розгортаються незалежно, дозволи доступу коригуються адміністраторами, а відомості про вразливості залежностей постійно оновлюються. Тому одноразове сканування не забезпечує актуальної оцінки стану безпеки.

Для вирішення цього завдання пропонується алгоритм безперервного моніторингу. Він поєднує періодичні перевірки з обробленням доступних подій зміни конфігурації.

Вхідними даними є реєстр контрольованих ресурсів, результати попередніх сканувань і повідомлення про зміни. На початковому етапі визначається перелік функцій, які потребують повторного аналізу.

Якщо отримано інформацію про розгортання нової версії, система ініціює перевірку відповідного програмного артефакту та конфігурації. Якщо змінилася політика доступу, доцільно повторити аналіз повноважень без обов'язкового сканування незмінного вихідного коду.

Окремим джерелом повторної перевірки є оновлення баз відомих вразливостей. Навіть якщо програмний артефакт не змінювався, раніше невідома вразливість його залежності може бути опублікована пізніше.

Для зменшення обчислювальних витрат доцільно використовувати інкрементальний аналіз. Якщо програмний артефакт має незмінний контрольний ідентифікатор, результати його статичного аналізу можуть використовуватися повторно. Водночас зіставлення залежностей з актуальними базами вразливостей потребує оновлення відповідних результатів.

Після кожної перевірки новий стан порівнюється з попереднім. Виявлені проблеми класифікуються як нові, збережені або потенційно усунені. Якщо певна знахідка відсутня в нових результатах через помилку сканування, її не можна автоматично вважати усунутою.

Результатом алгоритму є актуальна історія змін захищеності функцій та можливість визначення нових ризиків після оновлення застосунків.

Алгоритм керування життєвим циклом виявлених проблем

Для практичного використання системи необхідно не лише виявляти вразливості, а й контролювати їхній стан після передавання інформації відповідальним особам.

Для кожної знайденої проблеми передбачається визначення логічного стану. Початковий стан відповідає реєстрації результату перевірки. Після аналізу достовірності знахідка може бути підтверджена або позначена як така, що потребує уточнення.

Після підтвердження проблеми формується завдання на її усунення. Для вразливої бібліотеки це може бути оновлення залежності. Для небезпечної конфігурації необхідно змінити відповідну політику доступу. Для помилки прикладного коду може знадобитися модифікація програмної логіки.

Після внесення змін виконується повторна перевірка. Якщо відповідне порушення більше не виявляється за коректно виконаного аналізу, знахідка може переходити до стану підтвердженого усунення.

Окремо необхідно передбачити оброблення хибнопозитивних результатів. Якщо аналіз показав, що правило спрацювало помилково, відповідний запис може бути закритий із зазначенням обґрунтування. Таке рішення не повинно безумовно приховувати всі майбутні спрацювання аналогічного правила.

Для випадків, коли усунення неможливе в поточний момент, може застосовуватися контрольоване прийняття ризику. Воно повинно мати визначену підставу, відповідальну особу та обмежений період перегляду.

Результатом алгоритму є актуальний стан виявлених проблем, необхідний для контролю виконання заходів безпеки.

Алгоритм формування сповіщень і звітів

Результати моніторингу повинні бути представлені у формі, придатній для аналізу розробниками та адміністраторами хмарного середовища. Для цього передбачається алгоритм формування повідомлень про нові проблеми й зміни рівня ризику.

Вхідними даними є актуальний перелік знахідок, їхня пріоритетність та історія попередніх перевірок. Перед формуванням нового повідомлення система перевіряє наявність активного інциденту або завдання, пов'язаного з тією самою проблемою.

Якщо знахідка вже зареєстрована і її характеристики не змінилися, повторне повідомлення може не створюватися. Якщо критичність збільшилася або проблема з'явилася в новій версії застосунку, інформація оновлюється.

До повідомлення доцільно включати ідентифікатор функції, опис вразливості, рівень пріоритетності та відомості про джерело результату. Для

програмних залежностей зазначаються назва й уражена версія пакета. Для конфігураційних порушень визначається параметр, який не відповідає правилу безпеки.

Звіти можуть формуватися для окремої функції, застосунку або групи хмарних ресурсів. Для кожного об'єкта відображаються кількість активних проблем, їхній розподіл за критичністю та зміни порівняно з попереднім періодом.

Окреме значення має відображення повноти перевірок. Якщо частина функцій не була проаналізована через відсутність доступу до програмного артефакту або помилку API, це повинно бути зазначено у звіті. Інакше користувач може помилково сприймати відсутність знахідок як підтвердження безпечності всього застосунку.

Результатом алгоритму є структуровані повідомлення та звіти, які дозволяють визначати поточні проблеми та контролювати процес їхнього усунення.

Особливості програмної реалізації та масштабування

Програмна реалізація системи може передбачати центральний керувальний компонент, адаптери хмарних провайдерів, виконавці перевірок, сховище результатів та інтерфейс їхнього перегляду. Конкретний набір технологій визначається вимогами до підтримуваних платформ і доступними можливостями програмного середовища.

Керувальний компонент відповідає за планування перевірок і відстеження стану завдань. Адаптери забезпечують отримання конфігурацій і програмних артефактів через API відповідних хмарних платформ. Модулі аналізу виконують спеціалізовані перевірки та повертають результати в узгодженому форматі.

Для запобігання дублюванню перевірок кожне завдання повинно мати власний ідентифікатор. Якщо одна й та сама функція змінюється декілька разів протягом короткого періоду, система може об'єднувати або впорядковувати відповідні запити на перевірку.

Під час масштабування необхідно враховувати обмеження програмних інтерфейсів хмарних провайдерів. Надмірна кількість одночасних запитів може призводити до перевищення допустимої інтенсивності звернень. Тому доцільно застосовувати контрольоване обмеження паралелізму та повторні спроби з поступовим збільшенням інтервалу очікування.

Для аналізу великої кількості функцій завдання можуть розподілятися між незалежними виконавцями. При цьому необхідно зберігати зв'язок результатів із конкретним програмним артефактом і конфігурацією.

Повторне використання результатів незмінного статичного аналізу може зменшувати обчислювальні витрати. Однак висновки щодо актуальності відомих вразливостей повинні враховувати час останнього оновлення відповідної бази.

Важливим аспектом є захист самої системи моніторингу. Вона може отримувати службову інформацію про хмарні ресурси, конфігурації та вихідний код. Тому доступ до цих даних необхідно обмежувати відповідно до ролей користувачів.

Програмні артефакти, отримані для перевірки, не повинні виконуватися безпосередньо в середовищі керувального компонента. Якщо певний інструмент потребує виконання коду, необхідно використовувати спеціально ізольоване тестове середовище без доступу до робочих секретів та інших захищених ресурсів.

Алгоритм оброблення помилок і забезпечення достовірності результатів

Під час моніторингу можуть виникати помилки доступу до хмарного API, відсутність програмного артефакту, недоступність баз вразливостей або завершення часу виконання перевірки. Такі ситуації повинні оброблятися окремо від результатів аналізу безпеки.

Для кожного завдання визначається поточний стан виконання. Після створення воно переходить до стану очікування. Під час оброблення фіксується активне виконання. Після отримання результату завдання завершується успішно або переходить до стану помилки.

Якщо помилка пов'язана з тимчасовою недоступністю зовнішнього сервісу, може виконуватися повторна спроба. Якщо причиною є відсутність необхідних повноважень, повторення без зміни конфігурації зазвичай не усуває проблему.

У разі часткового успіху результати завершених перевірок повинні зберігатися. Наприклад, система може успішно проаналізувати конфігурацію функції, але не отримати її програмний артефакт. У такому випадку результат повинен містити висновки щодо конфігурації та окрему позначку про невиконаний аналіз залежностей.

Для збереження достовірності необхідно фіксувати джерело кожної знахідки, час перевірки та версію використаного правила. Це дозволяє відтворювати причини спрацювання після оновлення алгоритмів.

Окремо контролюється актуальність інформації. Результат перевірки попередньої версії функції не повинен без додаткового аналізу використовуватися як підтвердження стану нової версії.

Перевірка коректності алгоритмічного забезпечення

Для оцінювання функціонування запропонованих алгоритмів необхідно передбачити комплекс перевірок у контрольованому хмарному середовищі. На початковому етапі перевіряється правильність інвентаризації функцій і отримання їхніх конфігурацій.

Для цього може використовуватися тестовий набір безсерверних функцій із відомими характеристиками. Перевіряється повнота виявлення, правильність визначення версій і відповідність отриманих параметрів фактичній конфігурації.

Наступним етапом є перевірка аналізу залежностей. До тестових застосунків можуть включатися пакети з відомими опублікованими вразливостями. Результати зіставляються з очікуваними записами відповідної бази. Окремо перевіряється правильність оброблення транзитивних залежностей та випадків відсутності точної версії пакета.

Для перевірки аналізу конфігурації доцільно використовувати функції з різними політиками доступу. Частина компонентів повинна відповідати

встановленим правилам, а інші містити контрольовані порушення. Це дозволяє оцінювати правильність спрацювання алгоритмів і кількість хибнопозитивних результатів.

Алгоритм оцінювання пріоритетності перевіряється на наборах знахідок із різною технічною критичністю та умовами розміщення. Наприклад, порівнюються проблеми внутрішніх функцій і компонентів, доступних через зовнішні HTTP-інтерфейси. Отриманий порядок повинен відповідати встановленій політиці ризику.

Окремо перевіряється безперервний моніторинг. Для цього може виконуватися оновлення програмної версії, зміна повноважень та повторне розгортання функції. Система повинна виявляти відповідні зміни й оновлювати стан результатів.

Під час випробувань також необхідно оцінити вплив моніторингу на хмарну інфраструктуру. Контрольованими показниками можуть бути тривалість перевірки, кількість API-запитів, використання обчислювальних ресурсів і повнота отриманих результатів.

Для дослідження стійкості до помилок можуть моделюватися недоступність зовнішньої бази вразливостей, обмеження інтенсивності API-запитів і відсутність доступу до програмного артефакту. Очікуваною поведінкою є коректне визначення стану перевірки без формування необґрунтованого висновку про відсутність вразливостей.

Кількісні характеристики точності, продуктивності та ефективності алгоритмів можуть бути визначені після проведення відповідних експериментів. Доцільно окремо оцінювати повноту виявлення вразливостей, частку хибнопозитивних результатів і час отримання оновленої оцінки після зміни конфігурації. Розглянуте алгоритмічне забезпечення формує послідовний процес моніторингу безпеки безсерверних застосунків від інвентаризації хмарних ресурсів до оцінювання пріоритетності вразливостей та контролю їхнього усунення.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програмне забезпечення системи моніторингу та оцінювання вразливостей застосунків у безсерверних хмарних обчислювальних середовищах призначене для автоматизованого контролю захищеності програмних компонентів, аналізу їхніх залежностей і виявлення небезпечних конфігурацій.

Основними функціями є інвентаризація безсерверних застосунків, перевірка відомих вразливостей, оцінювання повноважень хмарних функцій та формування звітів про виявлені проблеми безпеки.

Система дозволяє організувати повторні перевірки після оновлення програмного коду або зміни параметрів хмарних ресурсів. Головне вікно програми показане на рисунку 5.1.



Рисунок 5.1 – Головне вікно ПЗ

Впровадження програмного комплексу передбачає підготовку середовища виконання, налаштування підключення до хмарних платформ і розгортання компонентів аналізу. Особливістю системи є використання програмних інтерфейсів хмарних провайдерів для отримання інформації про безсерверні функції. Такий підхід дозволяє виконувати перевірки без безпосереднього доступу до фізичних серверів, на яких працюють застосунки. Для поглибленого аналізу програмних залежностей додатково потрібний доступ до відповідних програмних артефактів або репозиторіїв.

Як робоче середовище можуть використовуватися серверні операційні системи Linux із підтримкою необхідних програмних залежностей. Конкретні вимоги до середовища виконання визначаються мовою програмування та бібліотеками, застосованими під час реалізації. Для розміщення основних компонентів можливе використання контейнеризації. Обсяг оперативної пам'яті, процесорні ресурси та дисковий простір визначаються кількістю контрольованих функцій, розміром програмних артефактів і кількістю одночасних перевірок.

Для організації взаємодії з хмарною інфраструктурою необхідно підготувати відповідні програмні адаптери. Залежно від обраної платформи можуть використовуватися API AWS Lambda, Azure Functions або Google Cloud Run functions. Кожний адаптер забезпечує отримання відомостей про функції, їхні версії, параметри виконання та конфігурації доступу. Підключення декількох провайдерів не є обов'язковою умовою роботи системи. Фактичний перелік підтримуваних платформ визначається реалізованими програмними модулями.

Підготовка підключення починається зі створення службової ідентичності з необхідними повноваженнями. Для моніторингу доцільно надавати переважно права на перегляд конфігурацій, списків ресурсів і метаданих програмних артефактів. Повноваження на зміну або видалення хмарних функцій не потрібні для звичайного аналізу захищеності. Облікові дані повинні зберігатися захищеним способом і не включатися безпосередньо до вихідного коду застосунку.

Наступним етапом є налаштування параметрів хмарного середовища. Для кожного підключення визначаються обліковий запис або проєкт, доступні регіони та перелік ресурсів, які підлягають перевірці. Окремо встановлюються обмеження інтенсивності звернень до API. Це необхідно для запобігання перевищенню квот хмарного провайдера під час одночасного опитування значної кількості функцій.

Для зберігання результатів моніторингу необхідно підготувати відповідне сховище даних. Воно повинно містити відомості про зареєстровані функції, результати перевірок, ідентифікатори вразливостей та історію змін. Конкретна система керування базами даних визначається програмною реалізацією. Важливо забезпечити зв'язок результатів аналізу з певною версією функції або ідентифікатором програмного артефакту. Це дозволяє відокремлювати проблеми поточної версії від вразливостей попередніх розгортань.

Для аналізу програмних залежностей може використовуватися формування Software Bill of Materials (SBOM). Як один із варіантів представлення складу програмного забезпечення розглядається CycloneDX. Отримані відомості про бібліотеки та їхні версії можуть зіставлятися з відкритими базами відомих вразливостей. Для цього доцільно налаштувати взаємодію з OSV API. Перед використанням зовнішнього джерела необхідно перевірити його доступність і можливість отримання актуальних даних.

Якщо програмна реалізація передбачає статичний аналіз вихідного коду, додатково налаштовуються відповідні засоби SAST. Для них визначаються підтримувані мови програмування, правила перевірки та доступ до вихідних файлів. Аналіз невідомих програмних артефактів доцільно виконувати в ізольованому середовищі. Звичайне сканування не повинно вимагати запуску недовіреного прикладного коду з повноваженнями керувального сервісу.

Після підготовки програмних залежностей налаштовується модуль оцінювання конфігурацій безпеки. Для нього визначаються правила перевірки публічної доступності функцій, параметрів автентифікації, наданих повноважень та використання секретів. Кожне правило повинно мати визначений ідентифікатор і опис очікуваної конфігурації. У разі відсутності необхідних

відомостей результат перевірки позначається як невизначений, а не як підтверджена відсутність порушення.

Окремо встановлюються параметри оцінювання критичності виявлених проблем. Для відомих вразливостей можуть використовуватися оцінки Common Vulnerability Scoring System (CVSS). Додатково враховуються доступність функції для зовнішніх запитів, її повноваження та значущість ресурсів, до яких вона звертається. Якщо передбачено використання Exploit Prediction Scoring System (EPSS), необхідно налаштувати отримання актуальних оцінок. Конкретні правила визначення пріоритетності повинні відповідати прийнятій політиці безпеки.

Рекомендована послідовність запуску починається з перевірки доступності сховища результатів і необхідних зовнішніх сервісів. Після цього активуються компоненти керування завданнями та адаптери хмарних платформ. Далі запускаються модулі аналізу залежностей, конфігурацій і повноважень. Після встановлення зв'язків між компонентами активуються засоби формування звітів та користувацький інтерфейс. Такий порядок дозволяє перевіряти працездатність окремих функціональних частин перед початком комплексного сканування.

Після запуску виконується первинна інвентаризація контрольованого хмарного середовища. Система отримує перелік доступних безсерверних функцій та їхні основні характеристики. Необхідно перевірити відповідність отриманого переліку фактичним ресурсам хмарного облікового запису. Додатково контролюється правильність визначення версій, регіонів розміщення та службових ідентичностей. Якщо частина ресурсів недоступна через обмеження прав, відповідна інформація повинна відображатися в результатах моніторингу.

Наступним етапом є виконання тестового сканування. Для цього доцільно використовувати контрольований безсерверний застосунок із відомим складом програмних залежностей і заздалегідь визначеними параметрами безпеки. Перевіряється правильність виявлення бібліотек, зіставлення їхніх версій із записами про вразливості та визначення небезпечних конфігурацій. Результати повинні містити відомості про джерело знахідки, відповідний програмний компонент і причину спрацювання правила.

Для перевірки аналізу повноважень можуть використовуватися тестові функції з різними політиками доступу. Необхідно переконатися, що система розрізняє обмежені та потенційно надмірні дозволи. Окремо перевіряється оброблення функцій із публічними HTTP-тригерами. Наявність зовнішньої точки доступу повинна оцінюватися з урахуванням її призначення та налаштованих механізмів автентифікації.

Після завершення первинних перевірок налаштовується періодичність моніторингу. Доцільно передбачити повторний аналіз після розгортання нової версії функції, зміни політики доступу або оновлення відомостей про вразливості. Для незмінних програмних артефактів можуть повторно використовуватися актуальні результати статичного аналізу. При цьому інформація про відомі вразливості залежностей повинна перевірятися відповідно до встановленої періодичності оновлення.

Під час експлуатації можливі помилки автентифікації, недоступність API хмарного провайдера, перевищення квот звернень або відсутність програмного артефакту. Для діагностики використовуються службові журнали керувального компонента та результати окремих завдань. Тимчасові мережеві помилки можуть оброблятися повторними спробами з обмеженою інтенсивністю. Якщо перевірка виконана лише частково, система повинна відображати неповноту результатів і не формувати необґрунтований висновок про відсутність вразливостей.

Перед введенням програмного комплексу в регулярну експлуатацію необхідно перевірити збереження результатів після перезапуску, правильність повторного сканування та формування повідомлень про нові проблеми. Додатково контролюються права доступу користувачів до результатів аналізу й захист службових облікових даних. Завершення відповідних перевірок дозволяє оцінити готовність системи до моніторингу безсерверних застосунків у визначеному хмарному середовищі.

Для ідентифікації програмного продукту передбачається надання відомостей про його розробника. Ця інформація використовується під час

супроводження системи та дозволяє встановити зв'язок програмного забезпечення з відповідною магістерською кваліфікаційною роботою.

Форма авторських даних системи моніторингу та оцінювання вразливостей безсерверних застосунків представлені на рисунку 5.2.

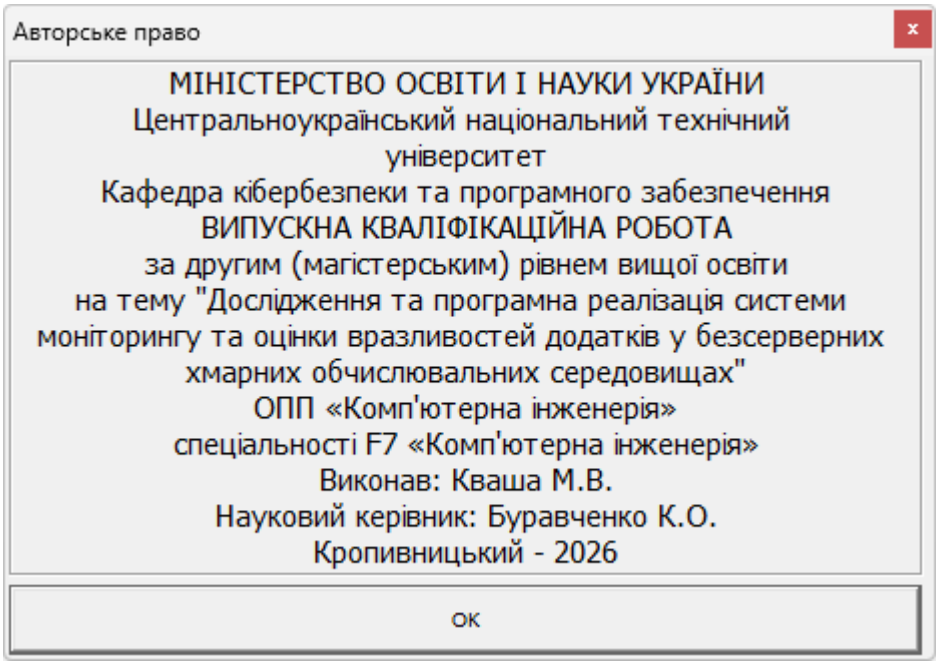


Рисунок 5.2 – Форма виконавця

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Метою розробки є дослідження та програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Об'єктом дослідження є процес моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Предметом дослідження є методи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Методи дослідження базуються на методах побудови комп'ютерних мереж, методів теорії харних обчислень, методів теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

– Розроблено вітчизняний продукт моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

7.1 Вступ

Охорона здоров'я працівників, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму повинна складати одну з головних завдань роботодавця.

Основою охорони праці є науковий аналіз умов праці, технологічних процесів, виробничого обладнання, робочих місць, трудових операцій, організації виробництва з метою виявлення шкідливих і небезпечних виробничих факторів, їх властивостей, особливостей виливу на організм людини. На підставі такого аналізу розробляються заходи та засоби, спрямовані на мінімізацію несприятливого впливу виробничих факторів, створення безпечних та нешкідливих умов праці.

Основою охорони праці є науковий аналіз умов праці, технологічних процесів, виробничого обладнання, робочих місць, трудових операцій, організації виробництва з метою виявлення шкідливих і небезпечних виробничих факторів, їх властивостей, особливостей виливу на організм людини. На підставі такого аналізу розробляються заходи та засоби, спрямовані на мінімізацію несприятливого впливу виробничих факторів, створення безпечних та нешкідливих умов праці

Для того, щоб об’єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам, необхідно здійснити санітарно-гігієнічну характеристику умов праці відділу, в якому працює програміст, над розробкою даного програмного продукту. У зв’язку з цим необхідно сконцентрувати увагу на небезпечних і шкідливих чинниках пов’язаних з постійною роботою за комп’ютером. Електробезпека є одним із критичних питань для співробітників, що працюють із технікою, яка одержує живлення з електричної мережі. При невиконанні норм електробезпеки можлива поразка електричним струмом.

7.2 Шкідливі і небезпечні фактори при роботі з комп’ютером

Електронно-обчислювальна машина (ПЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють програмісти, необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-97.

При роботі з використанням ПЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об’єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у тому числі високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат;

Відповідно до ст. 14 Закону «Про охорони праці» [1] на роботодавця покладено обов'язок забезпечити: безпеку працівників при експлуатації устаткування; застосування засобів індивідуального захисту працівників; відповідні вимоги охорони праці, умови праці на кожному робочому місці;

дотримання режиму праці та відпочинку працівників; навчання безпечним методам і прийомам виконання; інструктаж з охорони праці; організацію контролю над станом умов праці в робочих місць; проведення атестації робочих місць за умовами праці.

Максимально зменшити кількість шкідливих впливів на людину при високій продуктивності праці, створити комфортні умови для роботи людей – ось одна з головних задач охорони праці.

7.3 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [9].

Регулярне наочне знайомство персоналу із шляхами евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів

спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Оскільки при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

7.4 Пожежна безпека

Вимог до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об’єкти, цілі будівлі та їх частини, поверхи, на яких відповідальний співробітник повинен проводити такі організаційні роботи.

Відповідальні особи забов’язуються розробити, впровадити та підтримувати інструкції і положення на ввірених їм об’єктах, протипожежний режим відповідно до вимог, викладених в нормативних актах.

Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правил користування ними та утримання, наприклад:

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань пожежно-технічного мінімуму, у тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами.

Важливою складовою протипожежного режиму на будь-якому об’єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, в якому описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно для кожного об’єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань.

Детально про те, як розробити протипожежний режим, прописати порядки та інструкції, пояснюють на тематичних курсах і семінарах. [4]

Відповідно ДБН В 1.1-7-2016 «Пожежна безпека об’єктів будівництва» будинок можна віднести до II групи по ступені вогнестійкості й до категорії Д по ступені пожежонебезпеки.

Від розподільного щита по праву й ліву сторони встановлені кондиціонери, зовнішня електропроводка, поміщена в ізольований кабель. Висота проводки становить 2,2 м від рівня підлоги, її фіксація здійснюється за допомогою металевих кріплень. Біля кожного столу організований розподільний щит, розташований на текстолітовій пластинці, закріпленої на стіні на рівні 1м від підлоги. Усього до складу щита входять п'ять розеток і дві клеми заземлення.

Всі обчислювальні машини з'єднані із клемами заземлення. Чотири з п'яти розеток забезпечують подачу напруги 220 V, а одна, забезпечує подачу напруги в 36 В. Про це є відповідні написи на кожному розподільному щиті.

Робота обслуговуючого персоналу полягає в інсталяції необхідного програмного забезпечення й наступному його використанні в діалоговому режимі роботи з ЕОМ. Іноді може виникати необхідність написання допоміжних програм для поліпшення роботи вузла або для зниження витрат. З погляду забезпечення умов праці й вимог техніки безпеки для роботи програміста необхідно наступне:

- достатнє освітлення екрану дисплея й робочого місця;
- повна технічна справність устаткування, його електробезпечність;
- достатня пожежна безпечність приміщення;
- оптимальний мікроклімат, що сприяє продуктивній роботі;
- відповідність робочого місця вимогам ергономіки.

До небезпечних і шкідливих факторів, дії яких піддається програміст, можна віднести: можливість поразки електричним струмом, при електроні справності встаткування, порушенні заземлення або техніки безпеки; робота в мікрокліматі з неприпустимими параметрами; робота при недостатній освітленості екрана дисплея й робочого місця.

Відповідно НПАОП 40.1-1.21-98 “Правил безпечної експлуатації електроустановок споживачів” [6], приміщення можна віднести до приміщень без підвищеної небезпеки, оскільки це приміщення, сухе, з нормальною температурою й ізолюючими підлогами, що не має заземлених металоконструкцій.

Персональні ЕОМ можна віднести до першого класу електротехнічних виробів за способом захисту людини від поразки електричним струмом, оскільки їхні корпуси зроблені з ізолюючої пластмаси й кожен пристрій має заземлення. Відповідно до зазначених правил безпечної експлуатації електроустановок пристрої електроустановок ЕОМ можна віднести до електроустановок з робочою напругою до 1000 В.

Одною з достовірних причин пожежі в приміщенні з обчислювальною технікою може бути коротке замикання, що спричиняє спалах електропроводки. Для його попередження вся обчислювальна техніка, а також інші електричні пристрої повинні бути обладнані плавкими запобіжниками, а на вході електромережі повинен бути передбачений автомат захисту. Не слід користуватися електричними подовжувачами й трійниками, що не мають сертифікатів відповідності вимогам безпеки.

Необхідно передбачити наявність у межах досяжності первинних засобів гасіння пожежі (вогнегасників) для локалізації вогню власними засобами до приїзду команди пожежної охорони. Повинен бути розроблений план екстреної евакуації персоналу при виникненні загоряння. Кількість евакуаційних виходів повинне бути не менш двох. Допускається використання одного евакуаційного виходу, якщо відстань найбільш віддаленого робочого місця до цього виходу не перевищує 25 м.

7.5 Розрахункова частина

Запорукою безпечної роботи є виконання вимог електричної безпеки, оскільки все офісне обладнання заживлюється від електричної мережі. Одним з необхідних засобів електричної безпеки є встановлення захисного заземлення.

Початкові дані, необхідні для розрахунку захисного заземлення:

– допустимий опір розповсюдженню струму в землі від заземлю-вального пристрою $R_{зн} = 10$ Ом;

– питомий опір грунту в місці встановлення заземлювача $\rho_з = 100$ Ом/м;

– тип грунту – суглинок;

– тип заземлювача – труба, діаметром d=0.058 м і довжиною l =2.1 м;

– конструкція заземлювача – розташування заземлювачів по контуру. Розрахунок проводимо за стандартною методикою [7].

Визначимо розрахунковий опір землі:

$$\rho_{рз} = \phi \cdot \rho_З$$

де ϕ – коефіцієнт сезонності, що враховує коливання питомого опору при зміні вологості ґрунту протягом року; при використанні заземлювача довжиною l = 2.1 м при глибині закладання від вершини h = 0.4 м ϕ = 1.1 для четвертої кліматичної зони.

Схема розташування заземлювачів показана на рисунку 7.1.

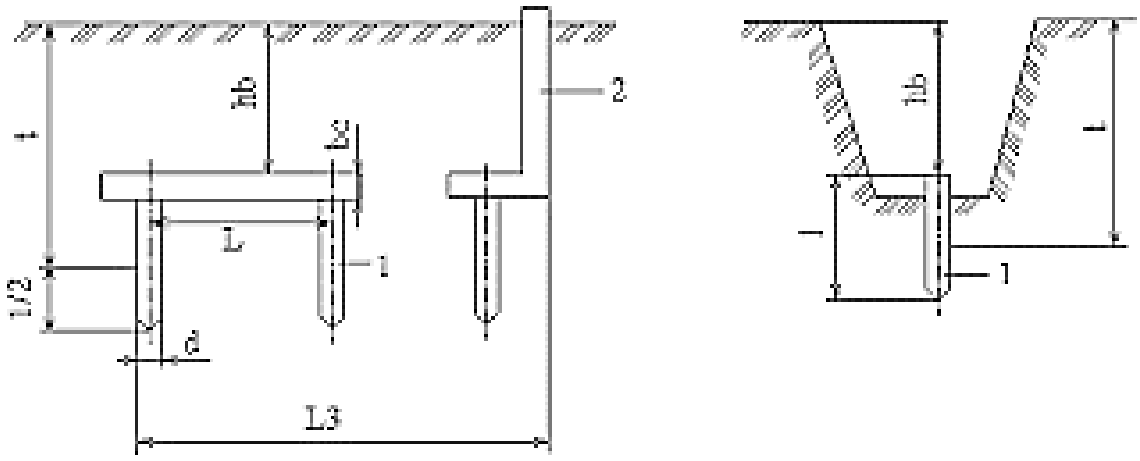


Рисунок 7.1 – Схема розташування заземлювачів

Опір землі:

$$\rho_{рз} = 1{,}1 \cdot 100 = 110 \text{ Ом·м}$$

Опір Rв, розповсюдженню струму в землі від одного вертикального заземлювача:

$$R_в = \frac{\rho_{рз}}{2\pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + 0.5\ln\frac{4t + l}{4t - l}\right)$$

де:

l – довжина заземлювача (l = 2.1 м);

d = 0.058 м – діаметр заземлювача при U<1 кВ та при S<100 кВА;

t – відстань від поверхні до середини заземлювача:

$$t = h + l/2 = 0.4 + 2.1/2 = 1.45 \text{ м.}$$

$$R_в = \frac{110}{2 \cdot 3.14 \cdot 2.1}\left(\ln\left(\frac{2 \cdot 2.1}{0.058}\right) + 0.5 \cdot \ln\left(\frac{4 \cdot 1.45 + 2.1}{4 \cdot 1.45 - 2.1}\right)\right) = 38.72 \text{ Ом}$$

Визначаємо потрібну кількість заземлювачів:

$$n' = \frac{R_{\text{в}}}{R_{\text{зн}}} = \frac{37.72}{10} = 3.9 \approx 4 \text{ шт.}$$

Коефіцієнт використання вертикальних заземлювачів враховує ефект екранування. При вибраному значенні k = a/l, де a – відстань між вертикальними заземлювачами, м; k = 1 при a = 1.5 м. Таким чином, коефіцієнт використання вертикального заземлювача за довідковими даними дорівнює $\eta_{\text{В}}$=0,6.

Кількість вертикальних заземлювачів з урахуванням коефіцієнту використання $\eta_{\text{В}}$ приблизно складає

$$n = \frac{R_{в}}{R_{зн} \cdot \eta_{в}} = \frac{38.72}{10 \cdot 0.6} = 6.45 \approx 7 \; шт.$$

Довжина горизонтального заземлювача, необхідна для розміщення вертикальних заземлювачів по контуру

$$L = a \cdot n = 1.5 \cdot 7 = 10.5 \text{ м}$$

Опір горизонтального заземлювача $R_{\text{Г}}$, Ом, прокладеного на глибині h=0.4м від поверхні землі буде

$$R_{г} = \frac{R_{рз}}{2 \cdot 3.14 \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot h} = \frac{110}{2 \cdot 3.14 \cdot 10.5} \cdot \ln \frac{2 \cdot 10.5^2}{0.058 \cdot 0.4} = 15.7 \text{ Ом}$$

де b = 0.04 м – ширина сталевої смуги, з якої виготовлений заземлювач.

Обчислюємо загальний опір:

$$R_{з} = \frac{R_{в} \cdot R_{г}}{n \cdot R_{г} \cdot \eta_{в} + R_{в} \cdot \eta_{г}} = \frac{38.72 \cdot 15.7}{6 \cdot 15.7 \cdot 0.6 + 38.72 \cdot 0.34} = 8.13 \text{ Ом}$$

де $\eta_{\text{Г}}$ – коефіцієнт використання горизонтального заземлювача ($\eta_{\text{Г}}$ = 0.34).

Маємо 7.13 Ом < 10 Ом, отже нормативне обмеження $R_{з} < R_{з.норм}$ виконується.

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз умов праці на робочому місці програміста, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливу шкоду та небезпеку, що можуть підстерігати його на робочому місці, та дотримання вимог нормативних актів з питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп’ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

8 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

– Досліджена система моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

– На основі отриманих результатів досліджень створена програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кваша М.В. Дослідження та програмна реалізація системи моніторингу та оцінки вразливостей додатків у безсерверних хмарних обчислювальних середовищах // Збірник праць молодих науковців ЦНТУ. – Вип. 17. – Кропивницький: ЦНТУ, 2027.
2. Vijay Kumar Velu. Mastering Kali Linux for Advanced Penetration Testing. Packt Publishing Ltd. 2022. 573 p.
3. Josh Armitage. Cloud Native Security Cookbook. O'Reilly Media. 2022. 516 p.
4. Massimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 p.
5. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.
6. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.
7. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.
8. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.
9. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.
10. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.
11. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p
12. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.
13. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

14. Богаченко, С., Смірнов, О., Буравченко, К., Смірнова, Т., Конопліцька-Слободенюк, О., Якименко, Н., Смірнов, С. «Обмеження ресурсів мобільних та edge-платформ при розробці програмного забезпечення для реалізації та оптимізації моделей машинного навчання». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 2026, т. 2 № 34, С. 677-686.

15. Вінтенко Б.Ю., Смірнова Т.В., Миронець І.В., Смірнов О.А., Буравченко К.О. «Метод оцінки функціональної стійкості комп'ютерно-орієнтованих процедур системи підтримки оперативного персоналу АЕС». *Системи управління, навігації та зв'язку*, 2026, вип. 2(84), С. 62-69.

16. Смірнова Т.В., Усік П.С., Лисенко І.А., Буравченко К.О., Смірнов О.А. «Методологія підтримки технологічних процесів у критичній інфраструктурі з забезпеченням безпеки інформації на основі хмарних технологій». *Безпека інформації.* 2025. Том 31 № 1. С. 49-60.

17. Смірнова Т.В., Буравченко К.О., Добринчук О.А., Смірнов С.А., Якименко Н.М., Смірнов О.А. «Хмарна технологія моніторингу ключових індикаторів ефективності технологічних процесів у критичній інфраструктурі». *Центральноукраїнський науковий вісник. Технічні науки.* 2025. № 12(43), ч. II. С. 36-54.

18. Вінтенко, Б.Ю., Миронець, І.В., Смірнов, О.А., Коваленко, О.В., Усік, П.С., Буравченко, К.О., Лисенко, І.А. «Логіко-структурна модель комп'ютерно-орієнтованої процедури системи підтримки оперативного персоналу АЕС». *Кібербезпека: освіта, наука, техніка*. 2025. Том 2 № 30. С. 413-427.

19. Vintenko, B., Myronets, I., Smirnov, O., Smirnova, T. «Enhancing the Reliability of Input Data for Nuclear Power Plant Operator Support Systems». *International Conference on Sensors, Devices and Systems. SDaS 2025. Lecture Notes in Electrical Engineering,* 2026. vol 1617. pp 3-12/

20. Kuznetsov O., Frontoni E., Arnesano M., Smirnov O., Khruskov B. «Biometric Authentication in TinyML: Opportunities and Challenges». *Tiny Machine Learning: Design Principles and Applications,* 2026, pp. 587-633.

21. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing,* 2025. pp. 21-61.

22. Chulinda L., Smirnov O., Shapenko L., Ustynova I., Bohatiuk I., Kelyp S. «The role of innovation in ensuring the safety of international civil aviation». *Ceur Workshop Proceedings,* 2025, 4024, pp. 530-542.

23. Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед та ін. «Системи інформаційної зброї та технології інформаційної війни»: підручник / Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед, Жарков Я.М., Смірнов О.А, Буравченко К.О., Давидюк А.В., Кононович В.Г., Корчинский В.В., Кудирко В.М., Фесенко А.О.; за заг. ред. В.М. Петрика, М.М. Присяжнюка.– К.: Видавничий центр "Кафедра", 2025.– 320 с.

24. Усік, П.С., Смірнова, Т.В., Буравченко, К.О., Смірнов, О.А., Улічев, О.С., Смірнов, С.А. «Дослідження технологій забезпечення кібербезпеки банківських систем з використанням штучного інтелекту». *Кібербезпека: освіта, наука, техніка.* 2025. Том 1 № 29. С.704–716, 2025

25. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

26. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security enhancement strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

27. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

28. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V.,

Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

29. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024, 3909, pp. 227–241.

30. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

31. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

32. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170-188.

33. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

34. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

35. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку

систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

36. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

37. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

38. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

39. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

40. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

41. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

42. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об’єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №*3*(19), 2023, С. 176-196.

43. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та*

перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)» м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

44. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

45. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

46. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

47. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

48. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

49. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

50. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої

функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

51. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

52. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

53. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

54. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) – 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

55. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

56. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

57. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I.,

Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

58. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

59. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.