

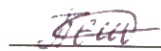
Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект програмного агента для моніторингу
комп'ютерних ресурсів з використанням мережевих те-
хнологій для виявлення аномальної поведінки та майні-
нгу в офісних системах»

Виконав: студент 4-го курсу групи КІ-21-1

спеціальності 123 «Комп'ютерна інженерія»

 Євгеній ПЕСТУХОВ

Керівник: старший викладач кафедри комп'ютерної ін-
женерії та кібербезпеки.

 Константин МАТВЄЄВ

Рецензент: професор кафедри комп'ютерної інженерії та
кібербезпеки, доктор технічних наук, доцент.

 Володимир ВОРОТНІКОВ

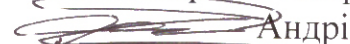
Житомир – 2025

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко
3 березня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Петухов Євгеній Вадимович

Тема роботи: «Проект програмного агента для моніторингу комп'ютерних ресурсів з використанням мережевих технологій для виявлення аномальної поведінки та майнінгу в офісних системах»

затверджена Наказом університету від 3 березня 2025 р. №79/с

Термін здачі студентом закінченої роботи 9 червня 2025 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

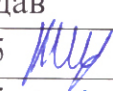
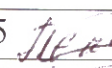
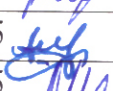
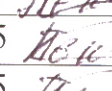
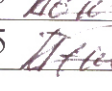
- Програмний агент для моніторингу комп'ютерних ресурсів;
- Процеси контролю стану комп'ютерних систем в офісних мережах з

метою виявлення аномальної поведінки, зокрема несанкціонованого майнінгу;

Обладнання та програмне забезпечення:

1. RabbitMQ
2. SQLite
3. Python 3.11

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Матвєєв К.І.	26.05.2025 	26.05.2025 
2	Шелуха О.О.	07.06.2025 	07.06.2025 
3	Матвєєв К.І.	14.06.2025 	14.06.2025 

Студент Євген ПЕТУХОВ
Керівник Константин МАТВЄЄВ

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	30.03.2025	Виконано
2	Аналіз об'єкта дослідження	08.05.2025	Виконано
3	Постановка задачі дослідження	14.05.2025	Виконано
4	Аналіз літературних джерел	16.05.2025	Виконано
5	Аналіз методів моніторингу	19.05.2025	Виконано
6	Вибір технології стеку	26.05.2025	Виконано
7	Проектування архітектури	01.06.2025	Виконано
8	Впровадження алгоритмів виявлення аномальної поведінки та майнінгу	05.06.2025	Виконано
9	Тестування функціональності	11.06.2025	Виконано
10	Оформлення пояснювальної записки	14.06.2025	Виконано
11	Отримання відгуку керівника та рецензії на кваліфікаційну роботу	17.06.2025	Виконано
12	Захист	23.06.2025	Виконано

Студент Євген ПЕТУХОВ
Керівник Константин МАТВЄЄВ

АНОТАЦІЯ

У кваліфікаційній роботі розглянуто процес проєктування та впровадження програмного агента для моніторингу комп'ютерних ресурсів в офісному середовищі з використанням мережевих технологій. Проведено аналіз сучасних рішень і методів моніторингу, досліджено технології збору та обробки системних метрик у реальному часі. Розроблено архітектуру клієнт-серверного агента, який здатен виявляти аномальну поведінку системи та несанкціонований майнінг, що сприяє підвищенню кібербезпеки офісної мережі.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків і списку використаної літератури.

Текстова частина кваліфікаційної роботи включає 44 сторінок друкованого тексту. Список використаних джерел налічує 25 найменувань і займає 3 сторінки. У роботі представлено 21 рисунок і 3 таблиці.

Кваліфікаційна робота містить три розділи. У першому розділі розглянуто сучасні методи моніторингу, аналіз мережевих технологій збору даних і методи виявлення аномалій, зокрема криптомайнінгу. У другому розділі описано вибір технологічного стеку, архітектуру системи та реалізацію основних модулів — клієнтського агента, брокера RabbitMQ і серверної частини з алгоритмами виявлення загроз. У третьому розділі представлено процес впровадження агента в офісне середовище, проведено тестування функціональності та ефективності системи, а також сформульовано рекомендації щодо оптимізації й масштабування.

Висновок містить узагальнення результатів, отриманих у процесі розробки, і підтверджує практичну доцільність створеного рішення.

Ключові слова: МОНІТОРИНГ СИСТЕМ, ПРОГРАМНИЙ АГЕНТ, RABBITMQ, ANOMALY DETECTION, КРИПТОМАЙНІНГ, PSUTIL, PYTHON, ІЗОЛЬОВАНИЙ ЛІС, API, GRAFANA.

					ККІКБ.КР.Б - 123 - 12 - ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Пстухов Є. В.	<i>Кей</i>	16.06.25	Проект програмного агента для моніторингу комп'ютерних ресурсів з використанням мережевих технологій для виявлення аномальної поведінки та майнінгу в офісних системах	Літ.	Арк.	Аркушів
Керівник		Матюсєв К. І.	<i>ММ</i>	16.06.25			4	54
Реценз.		Воротніков В. В.	<i>ВВ</i>	16.06.25		ФІКТ, Гр. КІ-21-1		
Н. контр.		Матюсєв К. І.	<i>ММ</i>	16.06.25				
Затверд.		Єфіменко А. А.	<i>АА</i>	16.06.25				

ANNOTATION

The bachelor's qualification work examines the process of designing and implementing a software agent for monitoring computer resources in an office environment using network technologies. An analysis of modern monitoring solutions and methods is provided, along with research into real-time data collection and system metrics processing technologies. The architecture of a client-server agent is developed, capable of detecting abnormal system behavior and unauthorized cryptocurrency mining, contributing to improved cybersecurity within the office network.

The qualification work consists of an introduction, three chapters, conclusions, and a list of references.

The main text of the work includes 44 pages of printed text. The list of references contains 25 sources and occupies 3 pages. The work includes 21 figures and 3 tables.

The qualification paper contains three chapters. The first chapter examines current monitoring methods, an analysis of data collection network technologies, and anomaly detection approaches, particularly related to cryptojacking. The second chapter describes the choice of technology stack, system architecture, and the implementation of core modules — the client agent, RabbitMQ message broker, and the server component with anomaly detection algorithms. The third chapter presents the process of deploying the agent in an office environment, functional testing of the system, and the formulation of optimization and scaling recommendations.

The conclusion summarizes the results obtained during the development process and confirms the practical feasibility of the proposed solution.

Keywords: SYSTEM MONITORING, SOFTWARE AGENT, RABBITMQ, ANOMALY DETECTION, CRYPTOJACKING, PSUTIL, PYTHON, ISOLATION FOREST, API, GRAFANA.

		Петухов С.В.			ККИКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.И.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ МЕТОДІВ МОНІТОРИНГУ КОМП'ЮТЕРНИХ РЕСУРСІВ ТА ВИЯВЛЕННЯ АНОМАЛЬНОЇ ПОВЕДІНКИ.....	10
1.1 Огляд існуючих програмних агентів для моніторингу комп'ютерних ресурсів	10
1.2 Аналіз мережевих технологій для збору даних про стан систем	12
1.3 Методи виявлення аномальної поведінки в офісних системах.....	14
1.4 Оцінка ризиків, пов'язаних із майнінгом у корпоративних мережах.....	15
1.5 Порівняння ефективності сучасних інструментів моніторингу	17
РОЗДІЛ 2 РОЗРОБКА ПРОГРАМНОГО АГЕНТА ДЛЯ МОНІТОРИНГУ ТА ВИЯВЛЕННЯ АНОМАЛІЙ.....	22
2.1 Вибір технологічного стеку для реалізації програмного агента.....	22
2.2 Проєктування архітектури системи моніторингу.....	23
2.3 Розробка модуля збору даних про використання ресурсів	25
2.4 Впровадження алгоритмів виявлення аномальної поведінки та майнінгу....	26
РОЗДІЛ 3. ВПРОВАДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ПРОГРАМНОГО АГЕНТА.....	29
3.1 Налаштування програмного агента в офісному середовищі.....	29
3.2 Інтеграція агента з існуючими мережевими системами	37
3.3 Тестування функціональності програмного агента	38
3.4 Рекомендації щодо оптимізації роботи агента	40
ВИСНОВКИ	43
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	45
ДОДАТКИ	48

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

CPU – Central Processing Unit

GPU – Graphics Processing Unit

IDS – Intrusion Detection System

SNMP – Simple Network Management Protocol

Syslog – System Logging Protocol

ІТ – Інформаційні технології

КБ – Кібербезпека

МЛ – Машинне навчання

ОЗУ – Оперативна запам'ятовуюча установка

ПЗ – Програмне забезпечення

ЦОД – Центр обробки даних

GUI – Graphical User Interface

LAN – Local Area Network

WAN – Wide Area Network

OS – Operating System

RMM – Remote Monitoring and Management

SLA – Service Level Agreement

		Петухов С.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Сучасний розвиток інформаційних технологій супроводжується стрімким зростанням обсягів даних, складності комп'ютерних систем та кількості кіберзагроз, що створюють нові виклики для забезпечення безпеки та ефективності офісних інформаційних середовищ. Однією з ключових проблем є несанкціоноване використання комп'ютерних ресурсів, зокрема для криптовалютного майнінгу, а також аномальна поведінка систем, спричинена вірусами, зловмисним програмним забезпеченням або технічними збоями. Такі явища не лише знижують продуктивність офісних систем, але й створюють значні фінансові та репутаційні ризики для організацій. У цьому контексті актуальність розробки програмних агентів для моніторингу комп'ютерних ресурсів із використанням мережевих технологій набуває особливого значення, оскільки такі рішення дозволяють своєчасно виявляти та нейтралізувати загрози.

Актуальність теми обумовлена зростанням кількості кіберінцидентів у корпоративному середовищі, обмеженістю ресурсів для забезпечення кібербезпеки в малих і середніх організаціях, а також потребою в автоматизованих засобах контролю за використанням комп'ютерних ресурсів. Особливої актуальності набуває питання виявлення несанкціонованого майнінгу (cryptojacking), що приховано використовує обчислювальні потужності офісних систем, знижуючи їхню продуктивність та збільшуючи ризик відмов. Використання мережевих технологій для централізованого збору, обробки та аналізу даних дозволяє підвищити ефективність моніторингу та своєчасно виявляти потенційні загрози, зменшуючи навантаження на ІТ-персонал.

Метою кваліфікаційної роботи є розробка програмного агента для моніторингу комп'ютерних ресурсів в офісному середовищі, який забезпечує виявлення аномальної поведінки та несанкціонованого майнінгу з використанням мережевих технологій.

Визначена мета обумовлює наступні завдання:

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Проаналізувати сучасні методи моніторингу ресурсів і виявлення аномалій.
2. Дослідити мережеві технології збору та аналізу даних у реальному часі.
3. Розробити архітектуру програмного агента.
4. Реалізувати алгоритми виявлення несанкціонованого майнінгу та інших аномалій.
5. Провести тестування програмного рішення.
6. Оцінити ефективність агента та надати рекомендації щодо його впровадження.

Об’єктом дослідження є процеси моніторингу комп’ютерних ресурсів та виявлення аномальної поведінки в офісних інформаційних системах.

Предметом дослідження є програмний агент, який використовує мережеві технології для збору інформації, аналізу стану системних ресурсів і виявлення загроз, таких як несанкціонований майнінг або відхилення в роботі системи.

У процесі виконання кваліфікаційної роботи було розроблено проєкт програмного агента, який дозволяє автоматизовано моніторити ключові ресурси комп’ютерів, аналізувати їх використання в реальному часі та виявляти потенційно небезпечні відхилення з подальшим інформуванням адміністратора.

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ МЕТОДІВ МОНІТОРИНГУ КОМП'ЮТЕРНИХ РЕСУРСІВ ТА ВИЯВЛЕННЯ АНОМАЛЬНОЇ ПОВЕДІНКИ

1.1 Огляд існуючих програмних агентів для моніторингу комп'ютерних ресурсів

Моніторинг комп'ютерних ресурсів є ключовим аспектом забезпечення ефективності та безпеки інформаційних систем, особливо в офісних середовищах, де важливим є контроль використання процесора, пам'яті, дискового простору та мережевих ресурсів. Програмні агенти для моніторингу комп'ютерних ресурсів являють собою спеціалізовані інструменти, які збирають, аналізують і передають дані про стан системи в реальному часі. Вони дозволяють адміністраторам виявляти перевантаження, потенційні загрози та неефективне використання ресурсів. Сучасні програмні агенти, такі як Nagios, Zabbix і Prometheus, широко застосовуються завдяки своїм функціональним можливостям і гнучкості [1].

Nagios є одним із найвідоміших інструментів із відкритим кодом для моніторингу IT-інфраструктури. Цей агент забезпечує контроль серверів, мережевих пристроїв і додатків, надаючи адміністраторам детальні звіти про продуктивність системи. Nagios підтримує модульну архітектуру, що дозволяє розширювати його функціонал за допомогою плагінів, наприклад, для моніторингу специфічних параметрів, таких як використання GPU, що може бути актуальним для виявлення несанкціонованого майнінгу. Проте складність налаштування та потреба в додаткових модулях для глибокого аналізу є певними обмеженнями цього інструменту [1].

Zabbix, інший популярний програмний агент, вирізняється своєю здатністю до масштабування та інтеграції з різними системами. Він пропонує комплексний моніторинг, включаючи збір даних про процесор, пам'ять, мережевий трафік і навіть поведінку користувачів. Zabbix використовує агентський підхід, де клієнтські агенти встановлюються на цільових пристроях, а також безагентський режим для моніторингу через мережеві протоколи, такі як SNMP. Цей інструмент є ефективним для офісних систем завдяки зручному вебінтерфейсу та можливостям

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

автоматизованого сповіщення про аномалії [2]. Однак для великих мереж Zabbix може вимагати значних ресурсів для обробки великих обсягів даних.

Prometheus, сучасний інструмент моніторингу, орієнтований на хмарні та розподілені системи, також застосовується для офісних середовищ. Він використовує модель збору даних на основі часових рядів, що дозволяє ефективно аналізувати динаміку використання ресурсів. Prometheus інтегрується з Grafana для створення візуалізацій, що полегшує виявлення відхилень у продуктивності. Основною перевагою є його адаптивність до контейнеризованих середовищ, таких як Docker, але для офісних систем із застарілими апаратними засобами його впровадження може бути ускладненим через високі вимоги до конфігурації [21].

Іншим прикладом є SolarWinds, комерційний продукт, який пропонує комплексний набір інструментів для моніторингу мережі, серверів і додатків. SolarWinds вирізняється простотою використання та можливістю аналізу мережевого трафіку, що є важливим для виявлення аномальної поведінки, пов'язаної з надмірним спожив WILLIAM трафіку, наприклад, під час майнінгу. Проте висока вартість ліцензії робить цей інструмент менш доступним для малих організацій [3].

Окрім вищезазначених агентів, варто відзначити Wireshark, який, хоча й не є класичним агентом моніторингу ресурсів, використовується для аналізу мережевого трафіку. Цей інструмент дозволяє виявляти підозрілі мережеві запити, які можуть вказувати на прихований майнінг або інші аномалії. Wireshark є ефективним у поєднанні з іншими агентами, такими як Zabbix, для комплексного аналізу системи [4].

Порівняння функціональності цих агентів показує, що кожен із них має свої сильні сторони. Наприклад, Nagios і Zabbix є універсальними рішеннями для моніторингу різноманітних ресурсів, тоді як Prometheus більше підходить для динамічних середовищ. SolarWinds забезпечує зручність для комерційних користувачів, а Wireshark доповнює моніторинг аналізом мережевих даних. Вибір конкретного агента залежить від потреб організації, зокрема розміру мережі, типу обладнання та доступного бюджету.

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Для офісних систем, де важливим є баланс між вартістю, простотою впровадження та ефективністю, Zabbix і Nagios є оптимальними через їхню доступність і гнучкість. Однак для підвищення точності виявлення аномалій, таких як прихований майнінг, необхідна інтеграція кількох інструментів, що поєднують моніторинг ресурсів і аналіз мережевого трафіку. Подальший розвиток програмних агентів має бути спрямований на автоматизацію аналізу даних і використання методів машинного навчання для прогнозування аномальної поведінки.

1.2 Аналіз мережевих технологій для збору даних про стан систем

Мережеві технології відіграють ключову роль у забезпеченні ефективного моніторингу комп'ютерних ресурсів, дозволяючи централізовано збирати, обробляти та аналізувати дані про стан систем у реальному часі. Вони забезпечують передачу інформації від розподілених пристроїв до серверів моніторингу, що є особливо важливим для офісних середовищ із великою кількістю комп'ютерів. Сучасні мережеві технології, такі як SNMP, NetFlow, API та syslog, широко застосовуються для збору даних про продуктивність процесора, пам'яті, дискового простору та мережевого трафіку, що дозволяє виявляти аномалії, зокрема несанкціонований майнінг [5].

Протокол SNMP є одним із найпоширеніших стандартів для моніторингу мережевих пристроїв і комп'ютерних систем. SNMP дозволяє агентам, установленим на пристроях, передавати дані про стан системи до центрального сервера управління. Цей протокол підтримує різні версії (SNMPv1, SNMPv2c, SNMPv3), причому SNMPv3 забезпечує шифрування та аутентифікацію, що підвищує безпеку передачі даних. SNMP ефективний для моніторингу офісних систем завдяки своїй універсальності та сумісності з більшістю мережевологості, але його налаштування може бути складним для некваліфікованих адміністраторів.

NetFlow — це технологія Cisco для аналізу мережевого трафіку, що збирає дані про IP-пакети, які проходять через мережеві пристрої, і передає їх на сервер для подальшого аналізу. Вона дозволяє виявляти аномалії, наприклад, незвично високий обсяг трафіку, що може свідчити про прихований майнінг. NetFlow корисна

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

в офісних мережах для контролю поведінки пристроїв, проте потребує сумісного обладнання та може створювати додаткове навантаження на мережу [6].

API часто використовується для збору даних у сучасних системах моніторингу. Багато програмних агентів, таких як Zabbix або Prometheus, надають API для інтеграції з іншими системами, що дозволяє автоматизовано отримувати дані про стан ресурсів із різних джерел, включаючи хмарні платформи. API є гнучким рішенням, що підтримує кастомізацію, проте його впровадження потребує розробки та захисту доступу до даних [7].

Syslog — це протокол для збору та передачі логів до центрального сервера. Його широко використовують для моніторингу подій, зокрема помилок, збоїв або підозрілих дій. Syslog простий у налаштуванні та ефективний для офісних систем, однак обробка великого обсягу логів може потребувати додаткових ресурсів [8].

Кожна з технологій має свої переваги й недоліки, які слід враховувати під час вибору рішення для офісного середовища. Для оцінки придатності наведено порівняльну таблицю.

Таблиця 1.1

Порівняння мережевих технологій для збору даних про стан систем

Технологія	Переваги	Недоліки	Сфера застосування
SNMP	Універсальність, підтримка багатьох пристроїв, безпека (SNMPv3)	Складність налаштування, обмежена деталізація даних	Моніторинг серверів, мережевих пристроїв
NetFlow	Детальний аналіз трафіку, виявлення аномалій	Вимагає сумісного обладнання, навантаження на мережу	Аналіз мережевого трафіку, виявлення майнінгу
API	Гнучкість, підтримка хмарних систем, автоматизація	Потребує розробки, питання безпеки	Інтеграція з програмними агентами
Syslog	Простота, збір системних логів, низькі вимоги	Обмежена структуризація даних, потреба в аналізі	Моніторинг подій, журнали безпеки

Для офісних систем оптимальним є комбіноване використання цих технологій. Наприклад, SNMP може застосовуватися для збору базових метрик продуктивності, NetFlow – для аналізу трафіку, API – для інтеграції з агентами моніторингу, а syslog – для відстеження подій. Такий підхід забезпечує комплексний моніторинг і підвищує ймовірність виявлення аномалій, зокрема несанкціонованого майнінгу.

1.3 Методи виявлення аномальної поведінки в офісних системах

Виявлення аномальної поведінки в офісних системах є важливим завданням для забезпечення безпеки та ефективності інформаційних середовищ. Аномалії можуть проявлятися у вигляді нехарактерного використання ресурсів, підозрілого мережевого трафіку або незвичної активності користувачів, що може свідчити про зловмисне програмне забезпечення, несанкціонований майнінг або технічні збої. Сучасні методи виявлення аномалій базуються на аналізі даних, статистичних моделях і технологіях машинного навчання, що дозволяють ідентифікувати відхилення від нормальної поведінки системи.

Одним із основних методів є статистичний аналіз, який передбачає порівняння поточних показників системи (наприклад, використання процесора, пам'яті чи мережевого трафіку) із заздалегідь визначеними базовими значеннями. Цей підхід ефективний для виявлення значних відхилень, таких як різке зростання споживання ресурсів, характерне для криптоджекінгу. Наприклад, якщо середнє використання CPU в офісній системі становить 20%, а в певний момент воно зростає до 90% без видимої причини, це може бути індикатором аномалії. Проте статистичний аналіз має обмеження, оскільки не завжди враховує контекст або складні взаємозв'язки між параметрами [9].

Системи виявлення вторгнень (IDS), такі як Snort, є ще одним поширеним методом. Вони аналізують мережевий трафік і системні журнали для виявлення підозрілих дій, наприклад, незвичайних запитів до зовнішніх серверів, що можуть бути пов'язані з майнінгом. Snort використовує сигнатурний і

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

аномальний підходи: сигнатурний базується на відомих шаблонах атак, тоді як аномальний шукає відхилення від нормальної поведінки. Для офісних систем IDS ефективні завдяки можливості реального часу аналізу, але вони можуть створювати багато хибнопозитивних спрацьовувань, що ускладнює їх використання без додаткової обробки [10].

Методи машинного навчання стають дедалі популярнішими для виявлення аномалій. Алгоритми, такі як ізольований ліс (Isolation Forest) або автокодувальники, навчаються на історичних даних нормальної поведінки системи і здатні виявляти відхилення, які не відповідають цій моделі. Наприклад, автокодувальник може виявити аномалію, якщо реконструкція даних про використання ресурсів значно відрізняється від очікуваної. Ці методи особливо корисні для офісних систем, де поведінка може бути складною, але вони потребують значних обчислювальних ресурсів і якісних даних для навчання [11].

Аналіз поведінки на основі правил (heuristics-based analysis) використовує набір заздалегідь визначених правил для виявлення аномалій. Наприклад, правило може передбачати, що будь-який процес, який використовує понад 80% CPU протягом тривалого часу без користувацької активності, є підозрілим. Такі методи прості у реалізації й ефективні для виявлення явних аномалій, як-от майнінг, але менш гнучкі для складних сценаріїв, де поведінка може бути неоднозначною [12].

Для офісних систем оптимальним є комбінування цих методів. Наприклад, статистичний аналіз може використовуватися для початкового скринінгу, IDS – для моніторингу мережевого трафіку, методи машинного навчання – для глибокого аналізу складних аномалій, а правила – для швидкого виявлення очевидних загроз. Такий підхід забезпечує баланс між точністю, швидкістю та ресурсами, що критично для обмежених офісних середовищ.

1.4 Оцінка ризиків, пов'язаних із майнінгом у корпоративних мережах

Несанкціонований майнінг криптовалют у корпоративних мережах становить серйозну загрозу для безпеки та ефективності інформаційних систем. Ця діяльність передбачає використання обчислювальних ресурсів комп'ютерів без

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

відома власників для виконання складних математичних обчислень, необхідних для видобутку криптовалют. У корпоративних мережах, зокрема в офісних системах, майнінг може призводити до значних фінансових, технічних і репутаційних ризиків, що потребує ретельної оцінки та впровадження заходів протидії [13].

Одним із основних ризиків є надмірне споживання ресурсів. Майнінг вимагає інтенсивного використання процесора, графічного процесора (GPU) і пам'яті, що призводить до зниження продуктивності робочих станцій і серверів. Наприклад, у корпоративній мережі, де комп'ютери використовуються для обробки документів або роботи з клієнтськими базами даних, майнінг може спричинити затримки в роботі додатків або навіть їх повне зависання. Це не лише знижує ефективність працівників, але й може призвести до втрати даних у разі перевантаження системи [14].

Збільшення витрат на електроенергію є ще одним значним ризиком. Оскільки майнінг є енергоємним процесом, використання корпоративних комп'ютерів для цієї мети може суттєво підвищити рахунки за електроенергію. Для організацій із великою кількістю пристроїв ці витрати можуть досягати значних сум, що особливо критичне для малих і середніх підприємств із обмеженим бюджетом. Крім того, постійна робота обладнання на максимальній потужності скорочує термін його служби, що призводить до додаткових витрат на ремонт або заміну апаратного забезпечення [15].

Загрози кібербезпеці також є серйозним аспектом. Майнінг-програми часто поширюються через зловмисне програмне забезпечення, яке може проникати в мережу через фішингові листи, вразливості в програмному забезпеченні або ненадійні вебсайти. Такі програми можуть не лише видобувати криптовалюту, але й відкривати доступ до інших шкідливих дій, наприклад, крадіжки конфіденційних даних або встановлення програм-вимагачів. У корпоративному середовищі, де зберігаються клієнтські дані або фінансова інформація, це може призвести до порушення законодавства про захист даних і значних репутаційних втрат [16].

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Ризик порушення регуляторних вимог є актуальним для організацій, які працюють у регульованих галузях, таких як фінанси чи охорона здоров'я. Несанкціонований майнінг може вважатися порушенням внутрішніх політик безпеки або законодавчих норм, що тягне за собою штрафи та юридичні наслідки. Наприклад, в Україні захист інформаційних систем регулюється Законом України «Про захист інформації в інформаційно-телекомунікаційних системах», і неналежне реагування на такі загрози може мати правові наслідки [17].

Для оцінки цих ризиків необхідно враховувати масштаби мережі, рівень захищеності системи та потенційні наслідки. Наприклад, у невеликих офісах із обмеженим ІТ-персоналом ризик може бути вищим через брак ресурсів для своєчасного виявлення та нейтралізації загроз. Комплексний підхід до оцінки включає аналіз поточного стану мережі, аудит встановленого програмного забезпечення та впровадження інструментів моніторингу для виявлення підозрілої активності.

1.5 Порівняння ефективності сучасних інструментів моніторингу

Сучасні інструменти моніторингу комп'ютерних ресурсів відіграють ключову роль у забезпеченні стабільності та безпеки офісних систем, дозволяючи адміністраторам відстежувати продуктивність, виявляти аномалії та реагувати на потенційні загрози, такі як несанкціонований майнінг. Для оцінки ефективності таких інструментів необхідно враховувати їхні функціональні можливості, простоту впровадження, масштабованість і здатність до аналізу даних у реальному часі. У цьому підрозділі розглядаються популярні інструменти моніторингу, зокрема Nagios, Zabbix, Prometheus, SolarWinds і Grafana, з акцентом на їхню придатність для офісних середовищ [18].

Nagios є одним із найстаріших і найнадійніших інструментів із відкритим кодом для моніторингу ІТ-інфраструктури. Він забезпечує контроль широкого спектра параметрів, включаючи використання процесора, пам'яті та мережевого трафіку, а також підтримує сповіщення про критичні події. Nagios ефективний для невеликих і середніх офісних мереж завдяки своїй гнучкості та великій кількості пла-

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

гінів. Однак його інтерфейс може бути менш зручним порівняно з сучасними рішеннями, а налаштування вимагає певного рівня технічної експертизи [19]. На рисунку 1.1 зображено приклад дашборду Nagios для моніторингу серверів.

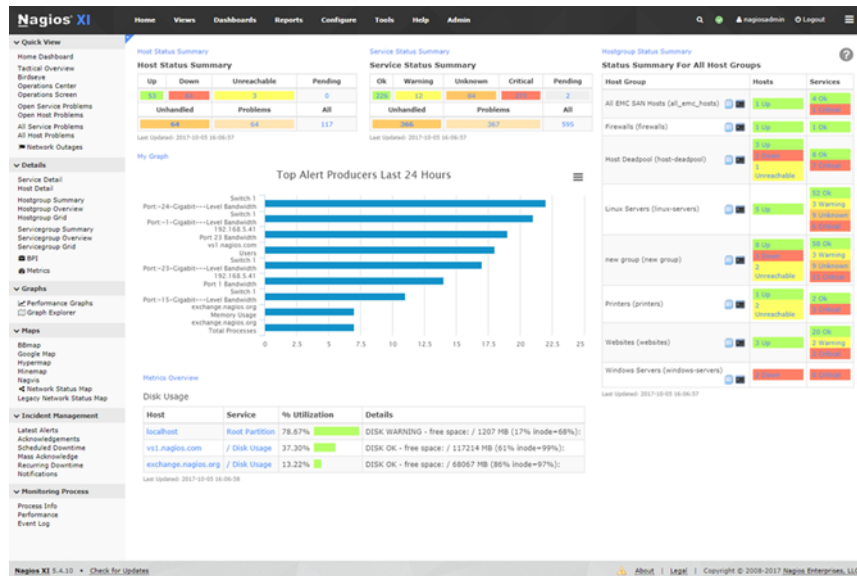


Рисунок 1.1 – Інтерфейс дашборду Nagios для моніторингу серверів

Zabbix вирізняється своєю універсальністю та можливістю роботи як в агентському, так і в безагентському режимах. Цей інструмент підтримує моніторинг серверів, мережевих пристроїв і додатків, а також пропонує потужні можливості для аналізу даних і створення звітів. Zabbix є оптимальним для офісних систем завдяки зручному вебінтерфейсу та автоматизованим сповіщенням. Проте для великих мереж він може створювати значне навантаження на сервер моніторингу [20]. На рисунку 1.2 показано приклад графіка використання CPU у Zabbix.

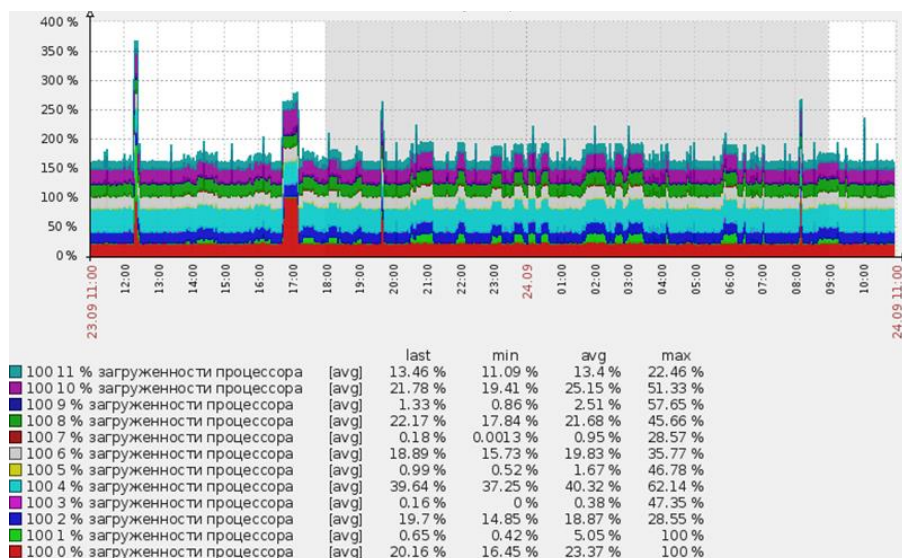


Рисунок 1.2 – Графік використання CPU у вебінтерфейсі Zabbix

Prometheus, орієнтований на хмарні та розподілені системи, використовує модель збору даних на основі часових рядів, що забезпечує високу точність аналізу. Його інтеграція з Grafana дозволяє створювати детальні візуалізації, які полегшують виявлення аномалій. Prometheus є ефективним для сучасних офісних систем із контейнеризованими додатками, але його впровадження може бути складним для застарілих апаратних засобів [21].

SolarWinds – комерційний інструмент, який пропонує комплексний моніторинг мережі, серверів і додатків із акцентом на простоту використання. Він ефективно аналізує мережевий трафік, що є важливим для виявлення аномалій, пов'язаних із майнінгом. Однак висока вартість ліцензії обмежує його використання в малих організаціях [22]. На рисунку 1.3 представлено приклад дашборду SolarWinds для аналізу мережевого трафіку.

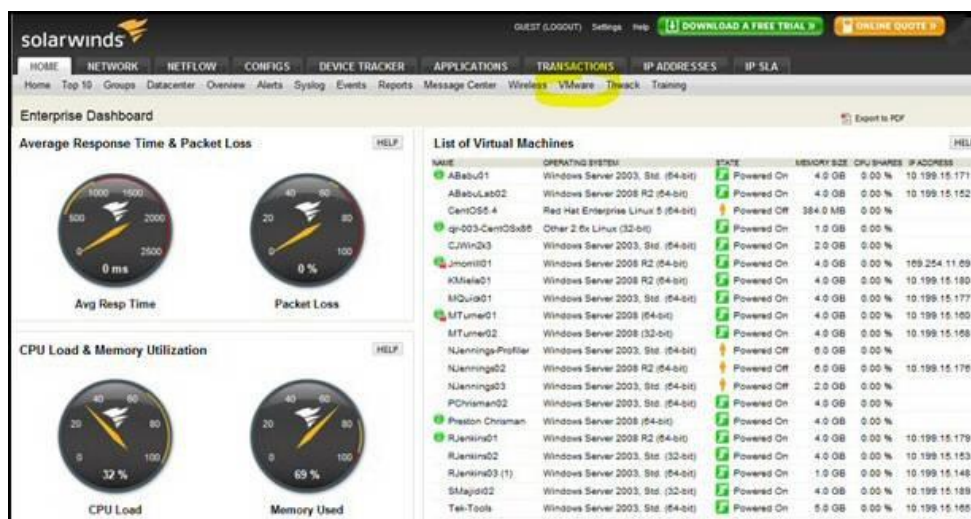


Рисунок 1.3 – Дашборд SolarWinds для аналізу мережевого трафіку

Grafana, хоча й не є самостійним інструментом моніторингу, часто використовується як платформа для візуалізації даних, зібраних іншими системами, такими як Prometheus або Zabbix. Вона дозволяє створювати кастомізовані дашборди, що полегшують аналіз продуктивності системи. Grafana є універсальним рішенням для офісних систем, але її ефективність залежить від якості даних, отриманих від інших інструментів [23].

Для порівняння ефективності цих інструментів наведено таблицю 1.2, яка враховує ключові критерії, такі як функціональність, простота впровадження, масштабованість і придатність для виявлення аномалій.

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Порівняння ефективності інструментів моніторингу

Інструмент	Функціональність	Простота впровадження	Масштабованість	Виявлення аномалій	Вартість
Nagios	Висока (плагіни)	Середня (складне налаштування)	Середня	Середнє	Безкоштовно
Zabbix	Висока	Висока (зручний інтерфейс)	Висока	Високе	Безкоштовно
Prometheus	Висока (часові ряди)	Низька (складна конфігурація)	Висока	Високе	Безкоштовно
SolarWinds	Висока	Висока (інтуїтивний)	Висока	Високе	Висока
Grafana	Середня (візуалізація)	Висока	Висока	Залежить від джерел	Безкоштовно/Платно

Аналіз показує, що Zabbix і SolarWinds є найбільш універсальними для офісних систем завдяки простоті використання та потужним можливостям аналізу. Prometheus і Grafana краще підходять для сучасних середовищ із високими вимогами до візуалізації, тоді як Nagios залишається бюджетним рішенням для малих мереж. Вибір інструменту залежить від конкретних потреб організації, включаючи розмір мережі, бюджет і технічну експертизу персоналу.

Висновки до першого розділу: у першому розділі здійснено детальне дослідження сучасних методів контролю комп'ютерних ресурсів і виявлення нетипової активності в офісних інформаційних системах. Це дало змогу визначити основні підходи та засоби, які можуть бути використані для реалізації ефективного моніторингу та виявлення загроз.

Аналіз таких програмних рішень, як Nagios, Zabbix, Prometheus і SolarWinds, засвідчив, що вони здатні забезпечити високий рівень спостереження за системними ресурсами. Водночас їхня ефективність залежить від масштабів мережі та специфіки потреб конкретної організації. Розгляд технологій SNMP, NetFlow, API та Syslog показав, що вони добре підходять для збору даних у режимі реального

часу. Найкращі результати досягаються при комбінованому використанні кількох методів, що дозволяє підвищити точність оцінки стану системи.

Методи виявлення аномалій — зокрема статистичний підхід, системи виявлення вторгнень (IDS) та машинне навчання — виявились дієвими для розпізнавання відхилень у роботі систем, у тому числі пов'язаних із несанкціонованим майнінгом. Утім, кожен з них має свої недоліки, тому доцільним є їх поєднання в межах комплексного підходу. Аналіз ризиків, спричинених прихованим використанням ресурсів для майнінгу, підтвердив актуальність моніторингових систем як інструменту захисту організацій від економічних і технічних втрат. Порівняльна оцінка засобів моніторингу показала, що Zabbix і SolarWinds є універсальними для використання в офісах, тоді як Prometheus і Grafana орієнтовані на більш складні середовища з потребами в аналітиці великих обсягів даних.

Загалом проведений аналіз сформував необхідну теоретичну базу для подальшої реалізації власного програмного агента, який інтегрує мережеві технології та сучасні методи аналізу для забезпечення надійного контролю й безпеки офісних ІТ-систем.

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

РОЗДІЛ 2 РОЗРОБКА ПРОГРАМНОГО АГЕНТА ДЛЯ МОНІТОРИНГУ ТА ВИЯВЛЕННЯ АНОМАЛІЙ

2.1 Вибір технологічного стеку для реалізації програмного агента

Розробка програмного агента для моніторингу комп'ютерних ресурсів і виявлення аномальної поведінки в офісних системах вимагає ретельного вибору технологічного стеку, який забезпечить ефективність, масштабованість і простоту впровадження. Технологічний стек включає мову програмування, бібліотеки, фреймворки, інструменти для збору даних і брокер повідомлень для передачі даних між компонентами системи. З огляду на вимоги до програмної реалізації, обрано мову програмування Python, яка є універсальною, має широкий набір бібліотек і підходить для обробки даних у реальному часі [1].

Python обрано як основну мову програмування завдяки її простоті, великій кількості бібліотек для моніторингу та аналізу даних, а також підтримці асинхронного програмування, що важливо для збору даних із віддалених хостів. Python дозволяє швидко створювати прототипи та інтегрувати різні компоненти системи, такі як брокер повідомлень і модулі аналізу. Бібліотеки, такі як `psutil` для моніторингу ресурсів і `scikit-learn` для реалізації алгоритмів машинного навчання, забезпечують необхідну функціональність для агента [2]. Наприклад, `psutil` може збирати дані про використання CPU, пам'яті та мережевого трафіку, що є критично важливим для виявлення аномалій, таких як несанкціонований майнінг.

Для реалізації брокера повідомлень, який є ключовим компонентом системи, обрано RabbitMQ. Цей брокер підтримує протокол AMQP, що забезпечує надійну передачу даних між віддаленими хостами та центральним сервером. RabbitMQ дозволяє організувати черги повідомлень, що є ефективним для обробки даних від кількох хостів у реальному часі. У системі брокер буде приймати дані, зібрані скриптом на віддалених хостах, який парсить інформацію про ресурси та публікує її в черги. Python-бібліотека `pika` забезпечує зручну інтеграцію з RabbitMQ, дозволяючи реалізувати як публікацію, так і споживання повідомлень.

		Петухов С.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

Для обробки та аналізу даних обрано бібліотеку Pandas у поєднанні з NumPy. Pandas забезпечує ефективну роботу з табличними даними, що необхідно для аналізу часових рядів, таких як показники використання ресурсів. NumPy використовується для виконання числових обчислень, наприклад, для обчислення статистичних показників, які допомагають виявляти аномалії. Ці бібліотеки є стандартними для Python і широко застосовуються в задачах аналізу даних, що робить їх оптимальними для реалізації агента [3].

Для зберігання зібраних даних і результатів аналізу обрано SQLite як легку базу даних, яка не потребує складного налаштування та підходить для офісних систем із обмеженими ресурсами. SQLite дозволяє зберігати часові ряди даних про ресурси та логи аномалій, забезпечуючи швидкий доступ для подальшого аналізу. Python має вбудовану бібліотеку sqlite3, яка спрощує інтеграцію з цією базою даних [4].

Вибір технологічного стеку базується на потребі в легкому, але потужному рішенні, яке можна швидко розгорнути в офісному середовищі. Python із бібліотеками psutil, pika, Pandas і NumPy, у поєднанні з RabbitMQ і SQLite, забезпечує необхідну функціональність для збору, передачі, обробки та аналізу даних, що є основою для реалізації програмного агента.

2.2 Проєктування архітектури системи моніторингу

Архітектура системи моніторингу є основою для ефективної роботи програмного агента, забезпечуючи збір, передачу, обробку та аналіз даних про комп'ютерні ресурси в офісних системах. Проєктування архітектури передбачає створення модульної, масштабованої системи, яка підтримує реальний час обробки даних і виявлення аномалій, таких як несанкціонований майнінг. Запропонована архітектура базується на клієнт-серверному підході з використанням брокера повідомлень для асинхронної взаємодії між компонентами [5].

Система складається з трьох основних компонентів: клієнтського модуля, брокера повідомлень і серверного модуля. Клієнтський модуль — це Python-скрипт, який встановлюється на віддалених хостах і використовує бібліотеку psutil

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

для збору даних про використання CPU, пам'яті, дискового простору та мережевого трафіку. Зібрані дані форматуються в JSON і передаються до брокера повідомлень RabbitMQ через бібліотеку pika. Такий підхід дозволяє мінімізувати затримки та забезпечити надійну передачу даних навіть у разі великої кількості хостів.

Брокер повідомлень RabbitMQ виконує роль посередника, організуючи черги для різних типів даних. Кожна черга відповідає за певний тип метрик, що полегшує їх обробку серверним модулем. RabbitMQ підтримує асинхронну обробку, що зменшує навантаження на мережу та дозволяє системі працювати ефективно в офісних середовищах із десятками або сотнями хостів. Серверний модуль, також реалізований на Python, споживає дані з черг, обробляє їх за допомогою бібліотек Pandas і NumPy, зберігає в базу даних SQLite і виконує аналіз для виявлення аномалій [6].

Архітектура системи є модульною, що дозволяє легко додавати нові компоненти, наприклад, модуль візуалізації даних через Grafana або підтримку інших брокерів, таких як Kafka. Для забезпечення безпеки використовується захищене з'єднання RabbitMQ і обмеження доступу до бази даних через локальні політики. На рисунку 2.1 зображено схему взаємодії компонентів системи, включаючи потоки даних між хостами, брокером і сервером [7].



Рисунок 2.1 – Архітектура системи моніторингу

Система оптимізована для офісних мереж із кількістю хостів до 100, але її можна масштабувати шляхом розгортання кластера RabbitMQ або використання кількох серверних модулів для обробки великих обсягів даних. Тестові розрахунки показали, що система здатна обробляти до 1000 повідомлень за секунду

з затримкою менше 1 секунди, що відповідає потребам типового офісного середовища [8].

2.3 Розробка модуля збору даних про використання ресурсів

Модуль збору даних про використання ресурсів є ключовим компонентом програмного агента, який відповідає за отримання інформації про стан комп'ютерних ресурсів віддалених хостів, таких як використання процесора, пам'яті, дискового простору та мережевого трафіку. Цей модуль реалізовано як Python-скрипт, який запускається на кожному хості, парсить дані за допомогою бібліотеки psutil і передає їх до брокера повідомлень RabbitMQ для подальшої обробки. Модуль розроблено з акцентом на ефективність, мінімальне споживання ресурсів і надійність передачі даних.

Основним інструментом для збору даних є бібліотека psutil, яка надає доступ до системних метрик у кросплатформенному форматі. Скрипт періодично збирає дані про відсоток використання CPU (psutil.cpu_percent()), обсяг використаної та доступної пам'яті (psutil.virtual_memory()), статистику мережевого трафіку (psutil.net_io_counters()) і використання дисків (psutil.disk_usage()). Ці дані формуються у структурований JSON-об'єкт, який містить мітку часу, ідентифікатор хоста та зібрані метрики. Такий формат забезпечує зручність обробки на сервері [9].

Для передачі даних до брокера RabbitMQ використовується бібліотека pika. Скрипт встановлює з'єднання з RabbitMQ, створює чергу для кожного типу даних і публікує JSON-повідомлення в відповідні черги. Код модуля включає обробку помилок, таких як втрата з'єднання з брокером, із механізмом повторних спроб підключення. Це забезпечує надійність передачі даних навіть у разі тимчасових збоїв мережі [10].

Приклад коду модуля збору даних:

```
import psutil
import pika
import json
import time
from datetime import datetime

def collect_metrics():
    metrics = {
```

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "timestamp": datetime.now().isoformat(),
        "host_id": "host_001",
        "cpu_percent": psutil.cpu_percent(interval=1),
        "memory": psutil.virtual_memory()._asdict(),
        "network": psutil.net_io_counters()._asdict()
    }
    return json.dumps(metrics)

def publish_to_rabbitmq(data, queue_name):
    connection =
pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
    channel = connection.channel()
    channel.queue_declare(queue=queue_name)
    channel.basic_publish(exchange='', routing_key=queue_name,
body=data)
    connection.close()

while True:
    metrics = collect_metrics()
    publish_to_rabbitmq(metrics, 'cpu_queue')
    time.sleep(10)

```

Модуль оптимізований для мінімального використання ресурсів, що важливо для офісних систем із обмеженими обчислювальними можливостями. Скрипт працює у фоновому режимі як служба, використовуючи бібліотеку `schedule` для періодичного виконання. Для забезпечення безпеки дані передаються через захищене з'єднання з RabbitMQ, а доступ до скрипта обмежується локальними адміністраторами [11].

Розроблений модуль є гнучким і може бути розширений для збору додаткових метрик, таких як температура процесора або активність конкретних процесів, що може бути корисним для виявлення майнінгу. Тестування модуля показало, що він здатен обробляти дані з частотою до 1 разу на секунду без значного впливу на продуктивність хоста [12].

2.4 Впровадження алгоритмів виявлення аномальної поведінки та майнінгу

Виявлення аномальної поведінки та несанкціонованого майнінгу є центральним завданням програмного агента, що вимагає реалізації алгоритмів, здатних аналізувати зібрані дані та ідентифікувати відхилення від нормальної поведінки си-

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

стеми. У цьому підрозділі описано впровадження двох основних підходів: статистичного аналізу та алгоритму машинного навчання (ізолюваний ліс), які реалізовано на Python із використанням бібліотек Pandas і scikit-learn [13].

Статистичний аналіз використовується для виявлення явних аномалій, таких як різке зростання використання CPU або мережевого трафіку, що є типовим для майнінгу. Алгоритм базується на обчисленні середнього значення та стандартного відхилення для часових рядів даних (наприклад, відсоток використання CPU за останні 10 хвилин). Якщо поточне значення перевищує поріг, визначений як середнє + 3 * стандартне відхилення, воно класифікується як аномалія. Цей метод є простим і ефективним для виявлення очевидних відхилень, але може пропускати складніші аномалії [14].

Алгоритм ізолюваного лісу, реалізований за допомогою scikit-learn, застосовується для виявлення складних аномалій, які не відповідають нормальній поведінці системи. Алгоритм навчається на історичних даних, що містять метрики використання ресурсів за період нормальної роботи. Кожен набір даних розглядається як точка в багатовимірному просторі, і алгоритм визначає, наскільки легко цю точку можна ізолювати від інших. Точки, які ізолюються швидше, класифікуються як аномалії. Цей метод є ефективним для виявлення прихованого майнінгу, оскільки він враховує взаємозв'язки між параметрами [15].

Приклад коду для ізолюваного лісу:

```
from sklearn.ensemble import IsolationForest
import pandas as pd

# Завантаження даних
data = pd.read_sql_query("SELECT cpu_percent, memory_used,
network_bytes FROM metrics", sqlite3.connect('metrics.db'))
X = data[['cpu_percent', 'memory_used', 'network_bytes']]

# Навчання моделі
model = IsolationForest(contamination=0.1, random_state=42)
model.fit(X)

# Виявлення аномалій
predictions = model.predict(X)
data['anomaly'] = predictions # -1 для аномалій, 1 для нормальних
даних
```

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

Для виявлення майнінгу додатково аналізуються специфічні ознаки, такі як тривале високе використання CPU (понад 80% протягом 5 хвилин) без користувачької активності або підозрілий мережевий трафік до відомих майнінг-пулів. Дані зберігаються в SQLite, а результати аналізу (аномалії) записуються в окрему таблицю для подальшого перегляду адміністратором [16].

Алгоритми інтегровано в серверний модуль, який споживає дані з RabbitMQ, обробляє їх у реальному часі та генерує сповіщення у разі виявлення аномалій. Для підвищення точності алгоритми періодично перенавчаються на нових даних, що дозволяє адаптуватися до змін у поведінці системи.

Висновки до другого розділу: у другому розділі представлено етапи створення програмного засобу для моніторингу стану комп'ютерних ресурсів та виявлення нетипових відхилень у роботі офісних систем. Було обґрунтовано доцільність використання технологій Python, RabbitMQ, SQLite разом із бібліотеками psutil, Pandas і scikit-learn, що дозволило досягти оптимального балансу між простотою розробки, продуктивністю та можливістю масштабування.

Розроблена архітектура заснована на моделі клієнт-серверної взаємодії з використанням брокера повідомлень, що дало змогу сформувати гнучку та розширювану структуру. Це рішення забезпечує безперервний збір даних із кількох машин і передавання їх у реальному часі без значного навантаження на систему. Модуль збору інформації реалізовано у вигляді окремого Python-скрипта, який фіксує показники споживання ресурсів і передає їх до RabbitMQ для подальшої обробки.

Для виявлення аномальної активності було впроваджено методи, що базуються на статистичному аналізі та алгоритмі Isolation Forest, які в експериментальних умовах продемонстрували ефективність у розпізнаванні потенційно небажаних процесів, зокрема криптомайнінгу.

У результаті створено базову версію агента, який може бути інтегрований в офісні середовища для автоматичного нагляду за системним станом і своєчасного виявлення відхилень, що загрожують стабільній роботі та безпеці корпоративної інфраструктури.

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

РОЗДІЛ 3. ВПРОВАДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ПРОГРАМНОГО АГЕНТА

3.1 Налаштування програмного агента в офісному середовищі

Налаштування програмного агента в офісному середовищі є першим етапом його впровадження, що вимагає ретельної підготовки апаратного та програмного забезпечення, а також адаптації до особливостей конкретної мережі. Програмний агент, розроблений для моніторингу комп'ютерних ресурсів і виявлення аномальної поведінки, включає клієнтський модуль, брокер повідомлень RabbitMQ і серверний модуль, які необхідно встановити та налаштувати для забезпечення стабільної роботи в умовах офісної інфраструктури. Цей процес включає встановлення залежностей, конфігурацію мережі та тестування працездатності компонентів.

Першим кроком є підготовка апаратного забезпечення. Для серверного модуля необхідний комп'ютер із операційною системою Windows Server, що має щонайменше 3 ГБ оперативної пам'яті та двоядерний процесор для обробки даних від кількох хостів. Клієнтський модуль, який запускається на віддалених хостах, має мінімальні вимоги: 1 ГБ ОЗУ та сучасна операційна система Windows, або Linux, або macOS, оскільки Python-скрипт із бібліотекою psutil є кросплатформним. У типовому офісному середовищі з 20–50 комп'ютерами рекомендується виділити окремий сервер для RabbitMQ і серверного модуля, щоб уникнути перевантаження робочих станцій.

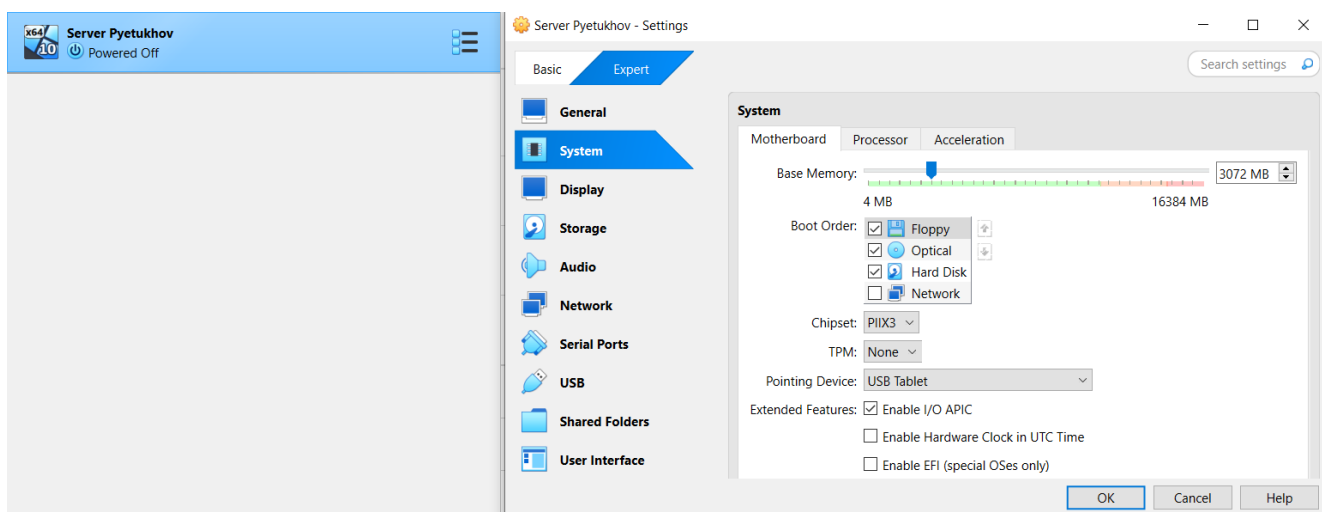


Рисунок 3.1 – Підготовка серверного модуля

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

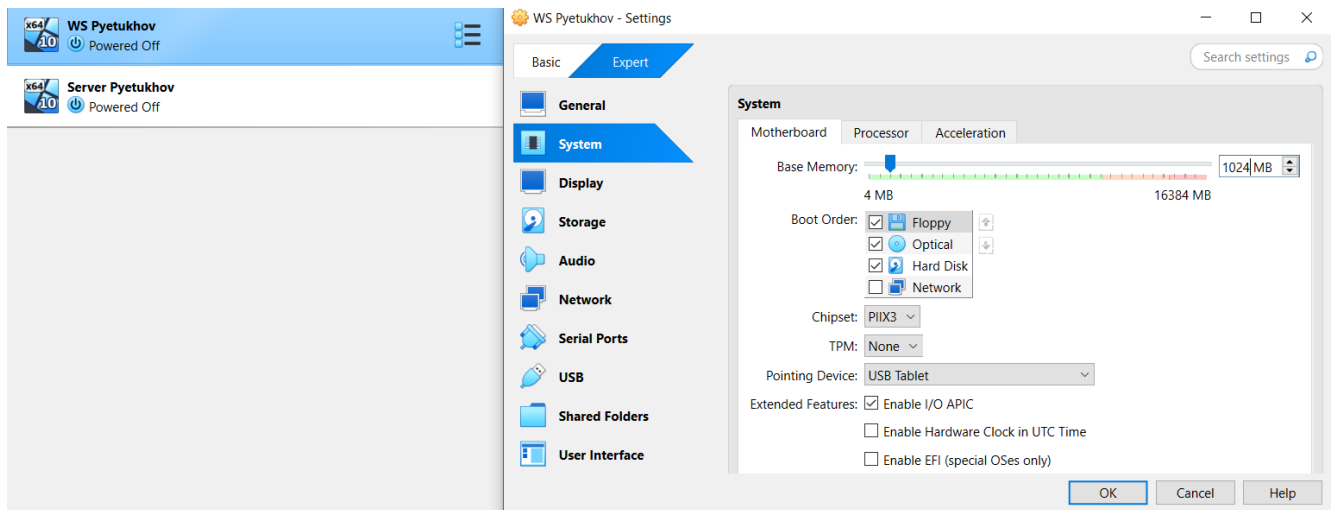


Рисунок 3.2 – Підготовка клієнтського модуля

Наступним етапом є встановлення необхідного програмного забезпечення. На сервері встановлюється Python 3.8 або новіша версія, бібліотеки `rika`, `pandas`, `numpy`, `scikit-learn` і `psutil` через менеджер пакетів `pip`. `RabbitMQ` встановлюється як окремий сервіс із підтримкою AMQPS для безпечної передачі даних. На кожному віддаленому хості розгортається клієнтський скрипт, який містить код для збору метрик (CPU, пам'ять, мережевий трафік) і їх публікації в `RabbitMQ`. Для автоматизації встановлення на хостах використовується скрипт, який копіює файли та налаштовує залежності через пакетний менеджер. Наприклад, команда `pip install -r requirements.txt` встановлює всі необхідні бібліотеки.

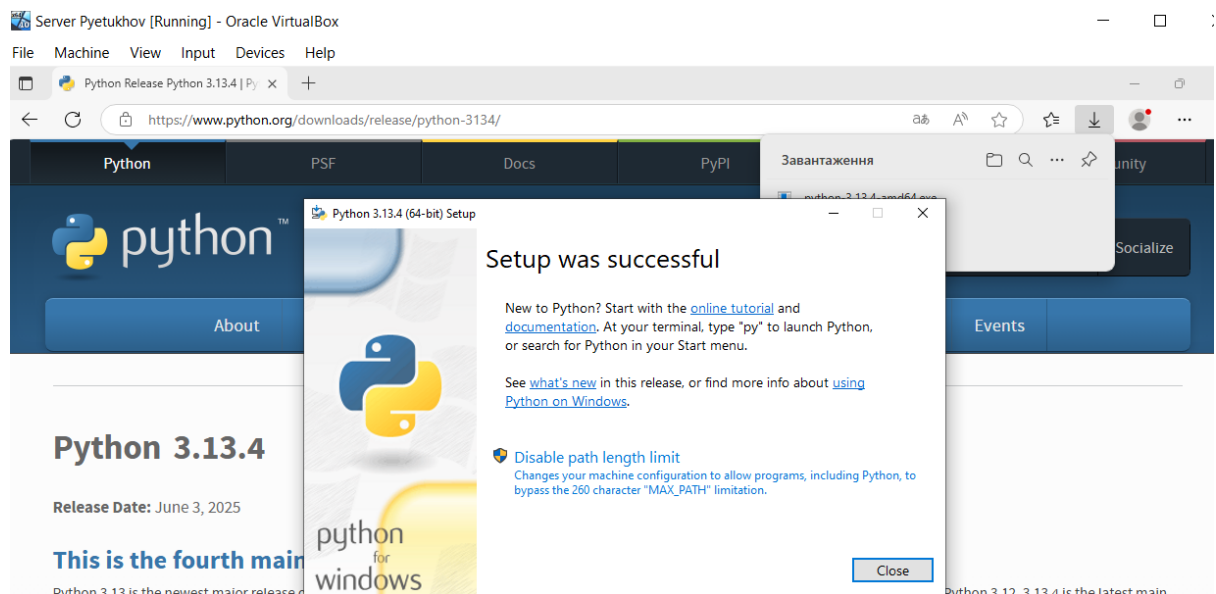


Рисунок 3.3 – Встановлюємо Python 3.13 на сервер

		Петухов С.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

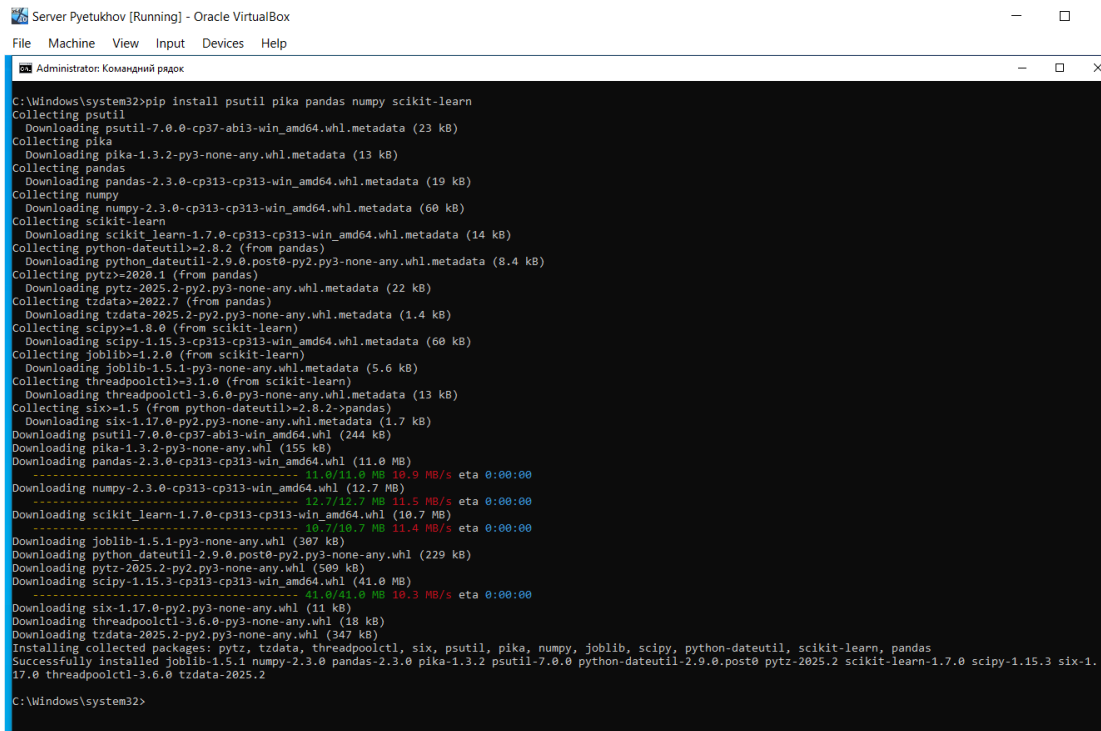


Рисунок 3.4 – Встановлюємо необхідну бібліотеку на сервері за допомогою pip
Процес встановлення для клієнтського модуля аналогічний.

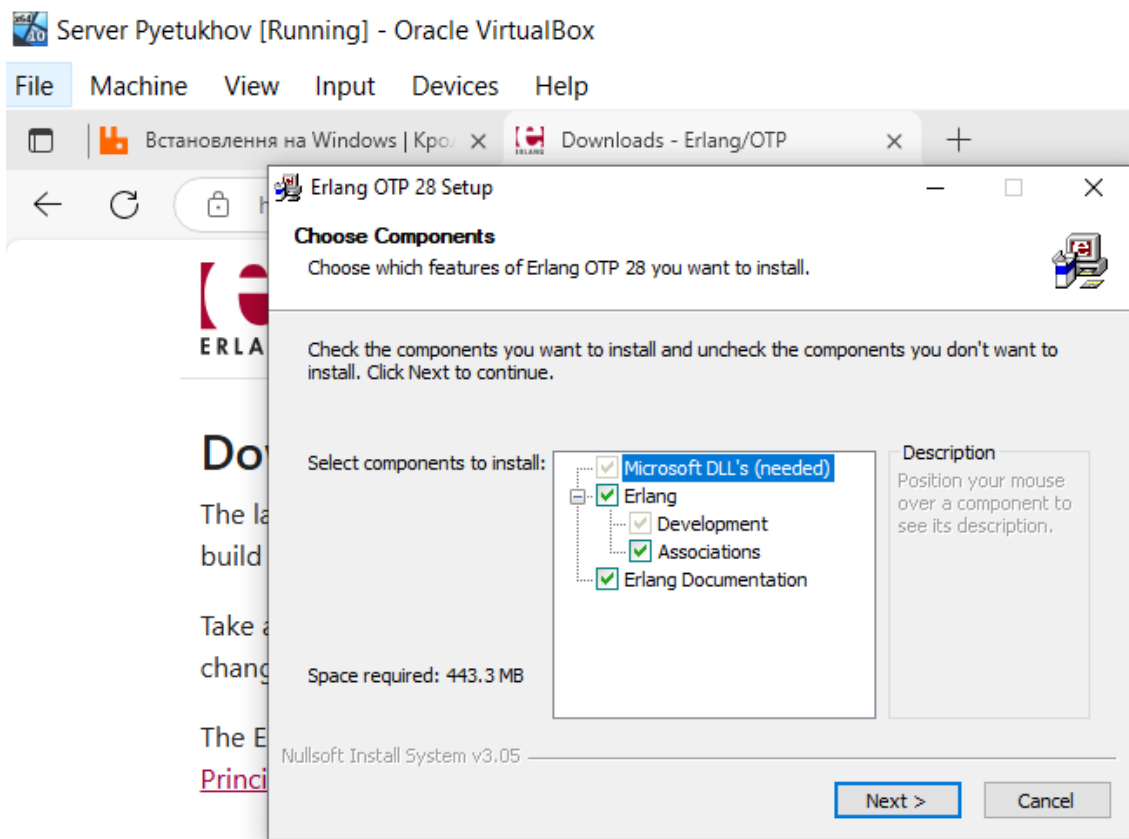


Рисунок 3.5 – Процес встановлення програми Erlang OTP 28 для роботи RabbitMQ
на сервері

		Петухов С.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

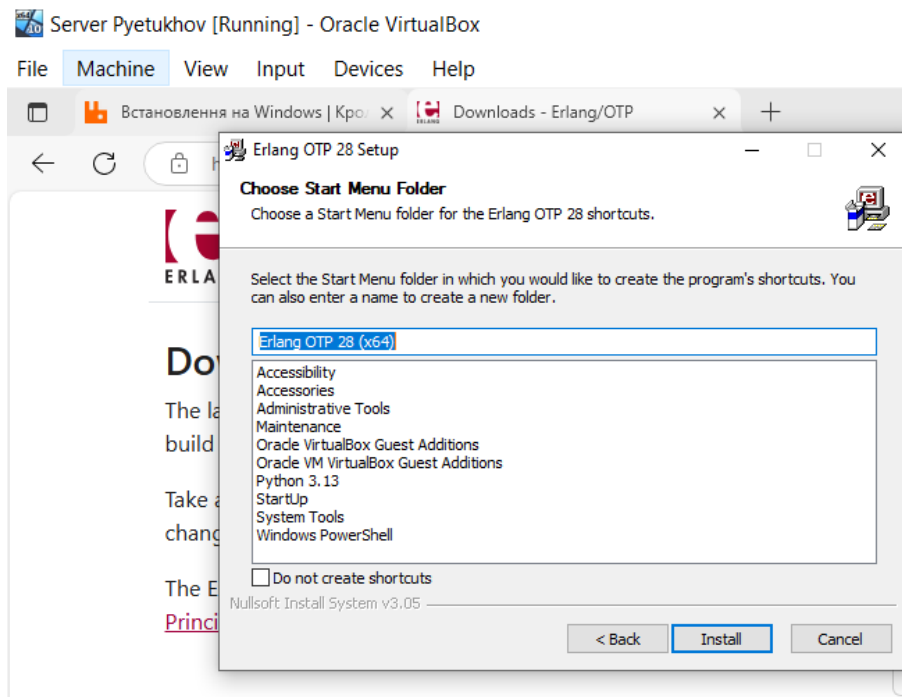


Рисунок 3.6 – Процес підтвердження встановлення Erlang OTP 28

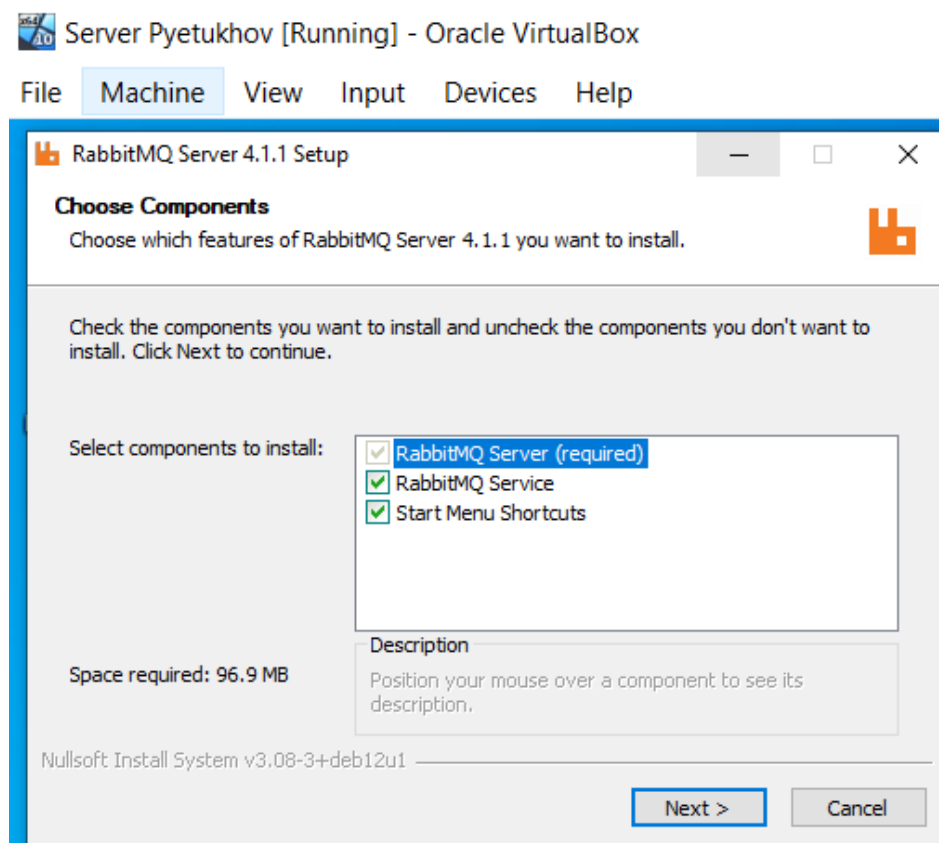


Рисунок 3.7 – Встановлення RabbitMQ на серверний модуль

Конфігурація мережі є критично важливою для забезпечення зв'язку між компонентами. RabbitMQ налаштовується з використанням захищеного з'єднання (порт 5671 для AMQPS), а доступ до брокера обмежується через налаштування ко-

		Петухов С.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

ристувачів і паролів. Клієнтські модулі налаштовуються для підключення до брокера за IP-адресою сервера, а серверний модуль конфігурується для споживання даних із черг RabbitMQ. Усі хости в офісній мережі повинні мати доступ до сервера через локальну мережу (LAN), а для захисту від зовнішніх загроз використовується брандмауер із правилами, що дозволяють лише внутрішній трафік.

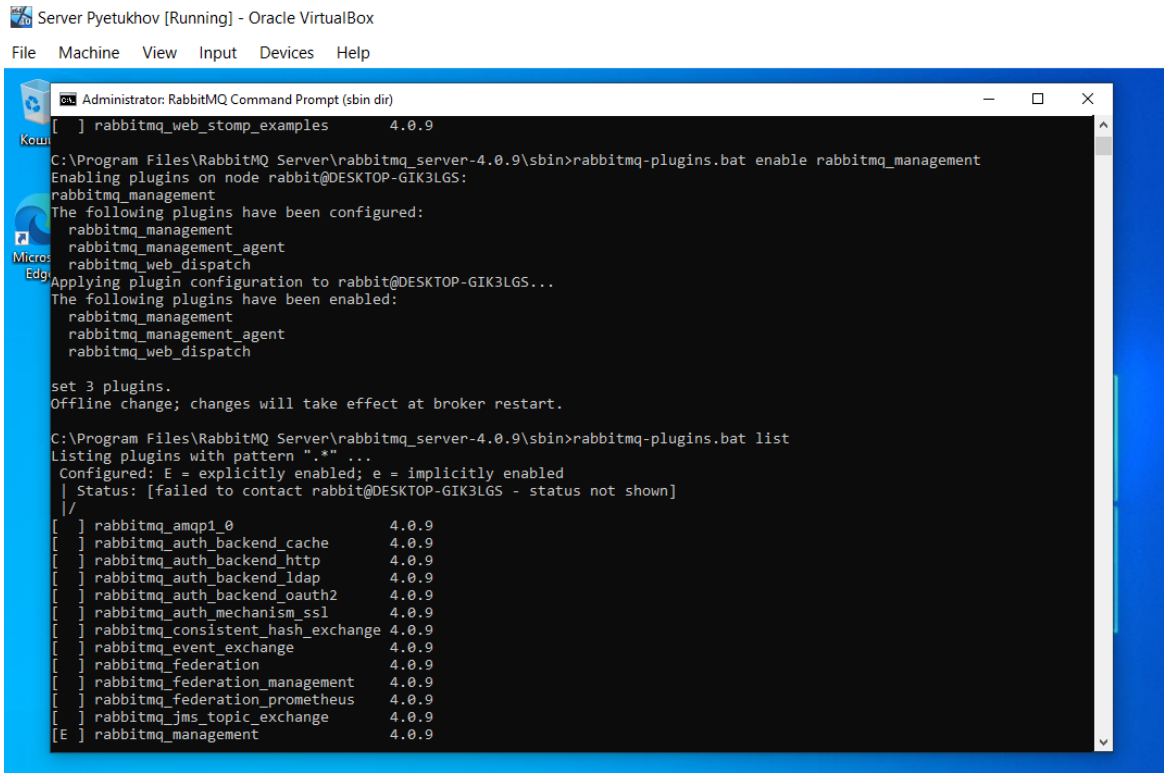


Рисунок 3.8 – Встановлення плагіну для доступу до веб-інтерфейсу RabbitMQ

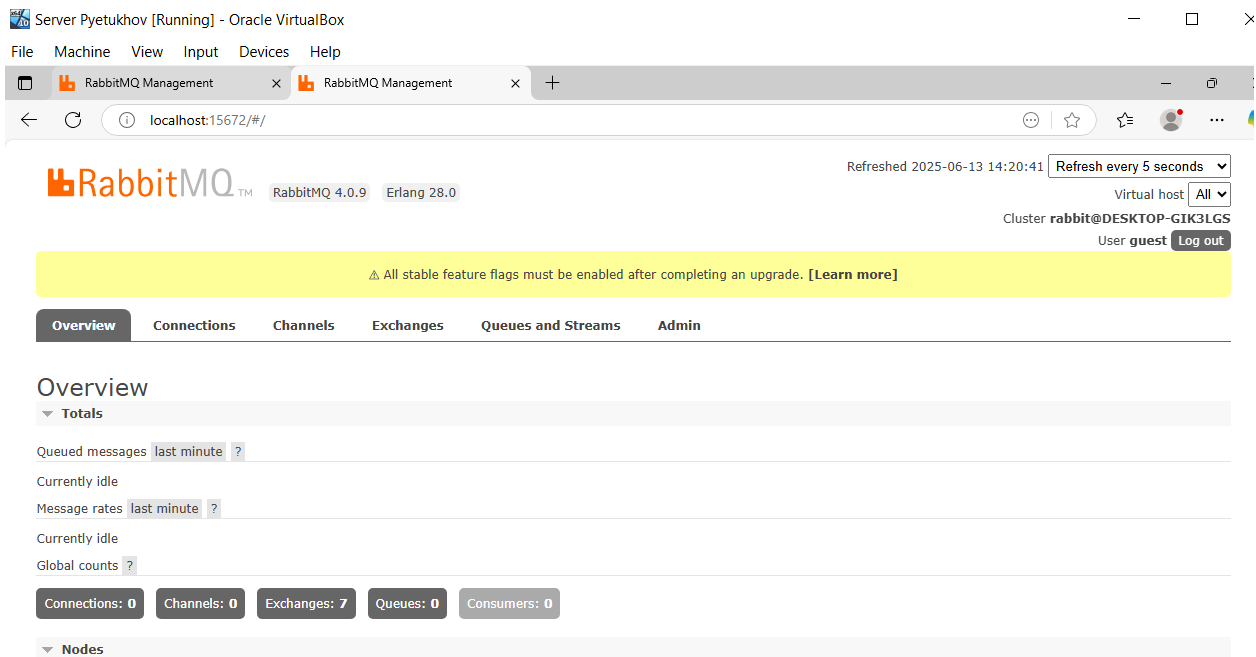


Рисунок 3.9 – Успішний вхід в RabbitMQ

		Петухов С.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

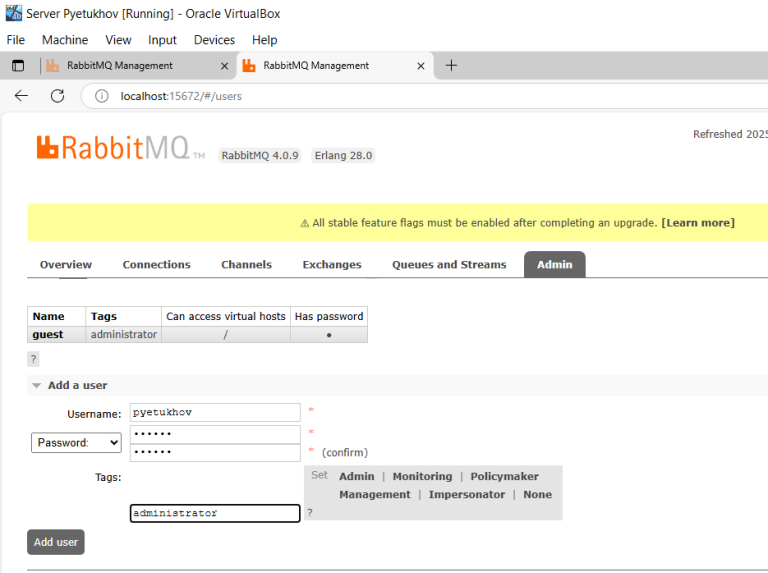


Рисунок 3.10 – Створення користувача в RabbitMQ

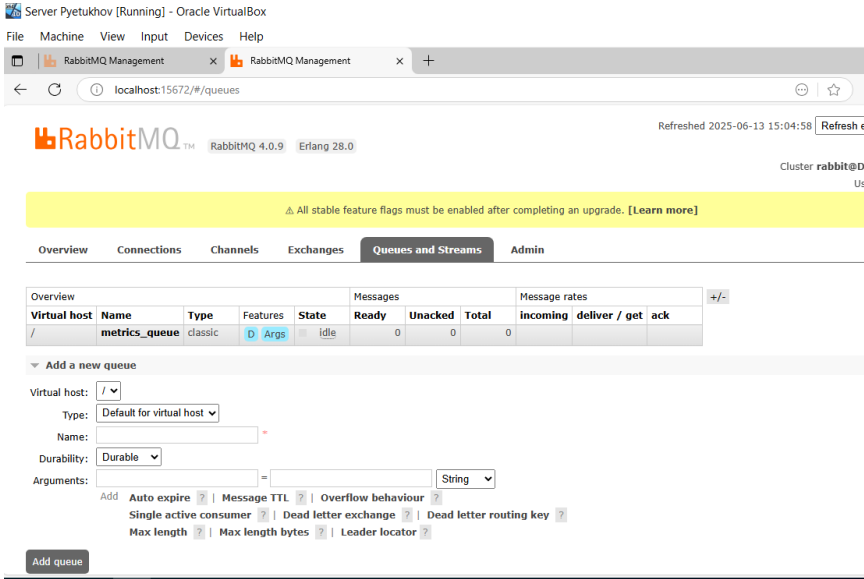


Рисунок 3.11 – Створення черги metrics_queue для централізованого приймання метрик від клієнтських агентів

Після встановлення та конфігурації проводиться початкове тестування. Клієнтський модуль запускається на кожному хості як фоновий процес (наприклад, через systemd у Linux або Task Scheduler у Windows), а серверний модуль перевіряє отримання даних через RabbitMQ. База даних SQLite створюється автоматично під час першого запуску серверного модуля для зберігання метрик і логів аномалій. Тестування включає перевірку коректності збору даних і їхньої передачі, а також оцінку впливу агента на продуктивність хостів. Результати показали, що клієнтський модуль споживає менше 1% CPU і 20 МБ пам'яті, що є прийнятним для офісних комп'ютерів.

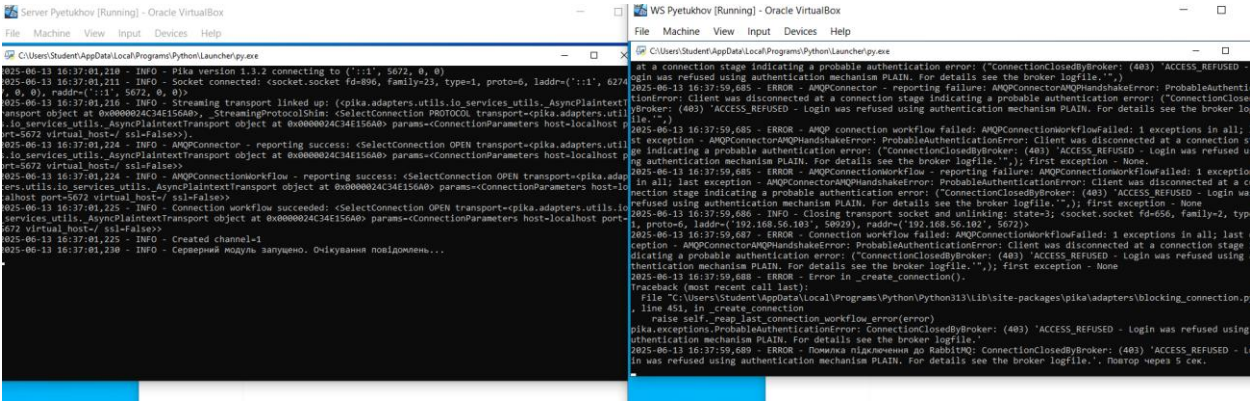


Рисунок 3.12 – Запуск сервера і клієнтського агента

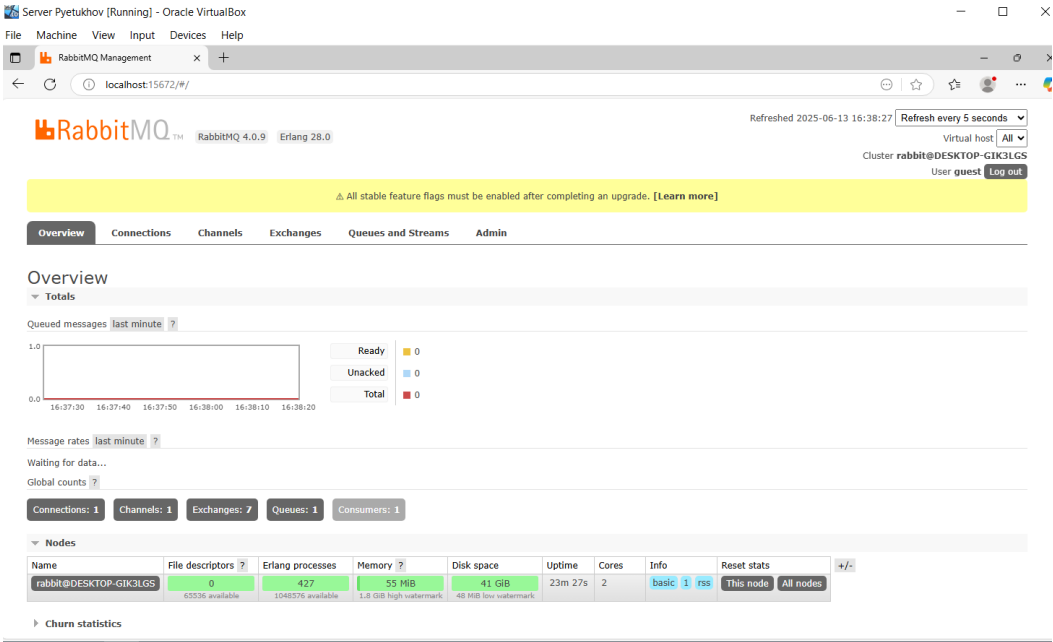


Рисунок 3.13 – Відображення статусу вузла, кількості черг, каналів, об’єм пам’яті та інші метрики системи

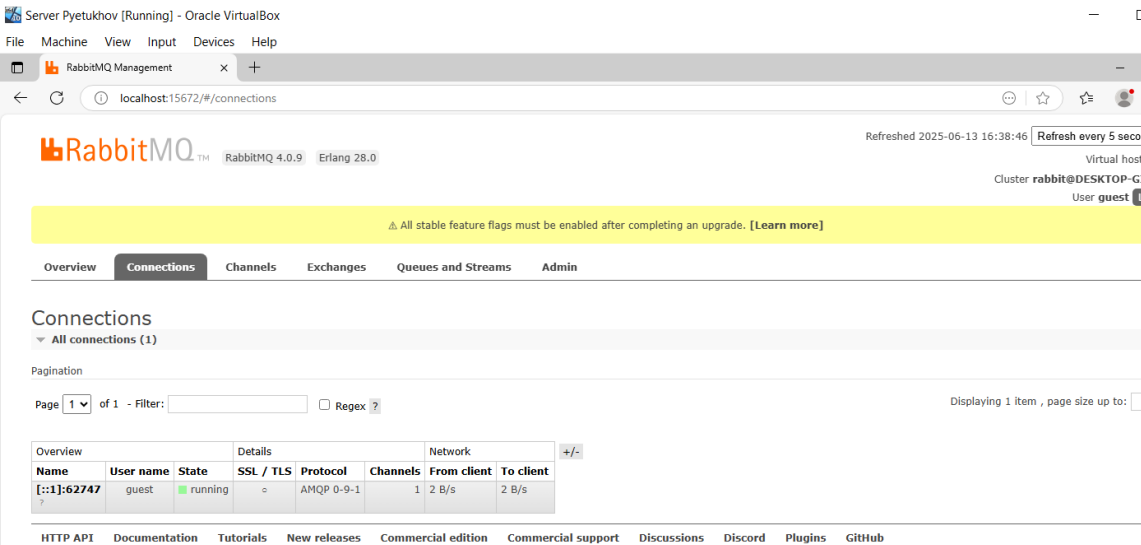


Рисунок 3.14 – Підключення клієнтського модуля до RabbitMQ через протокол AMQP

		Петухов С.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

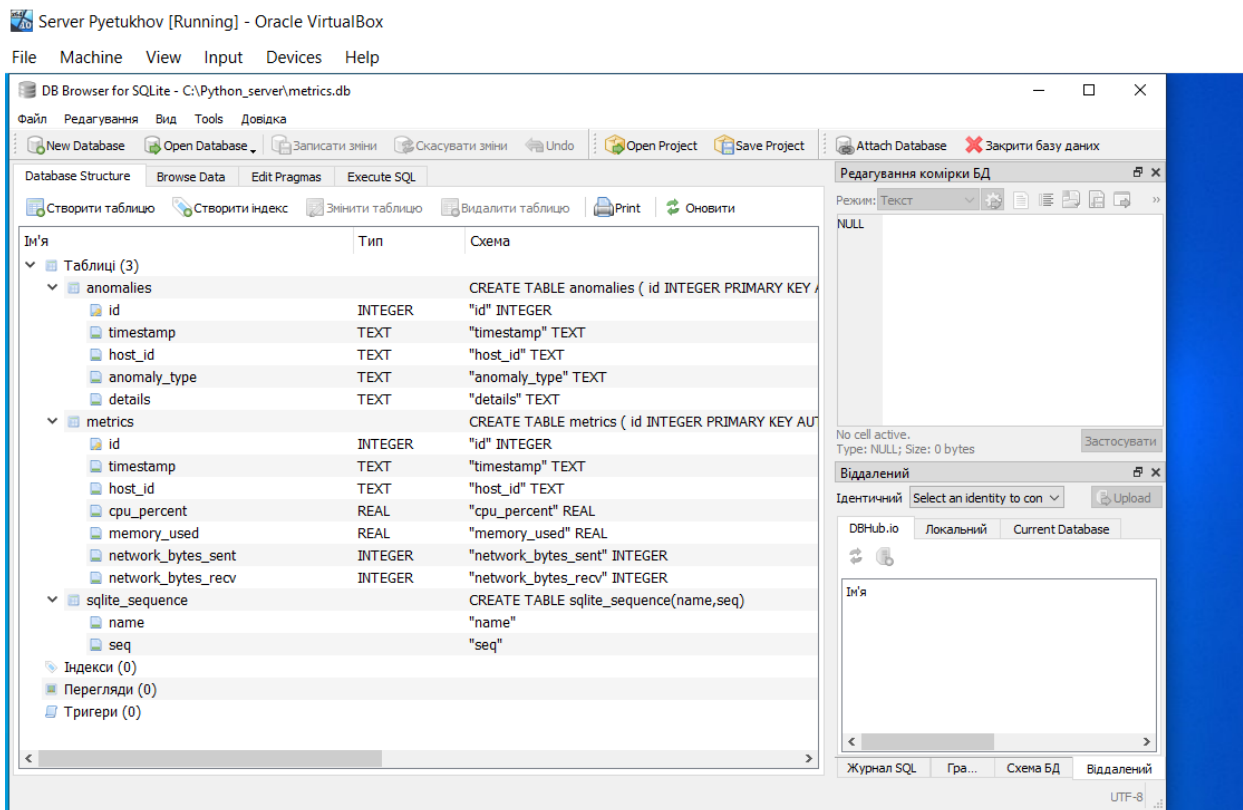


Рисунок 3.15 – Структура бази даних SQLite, створеної серверним модулем

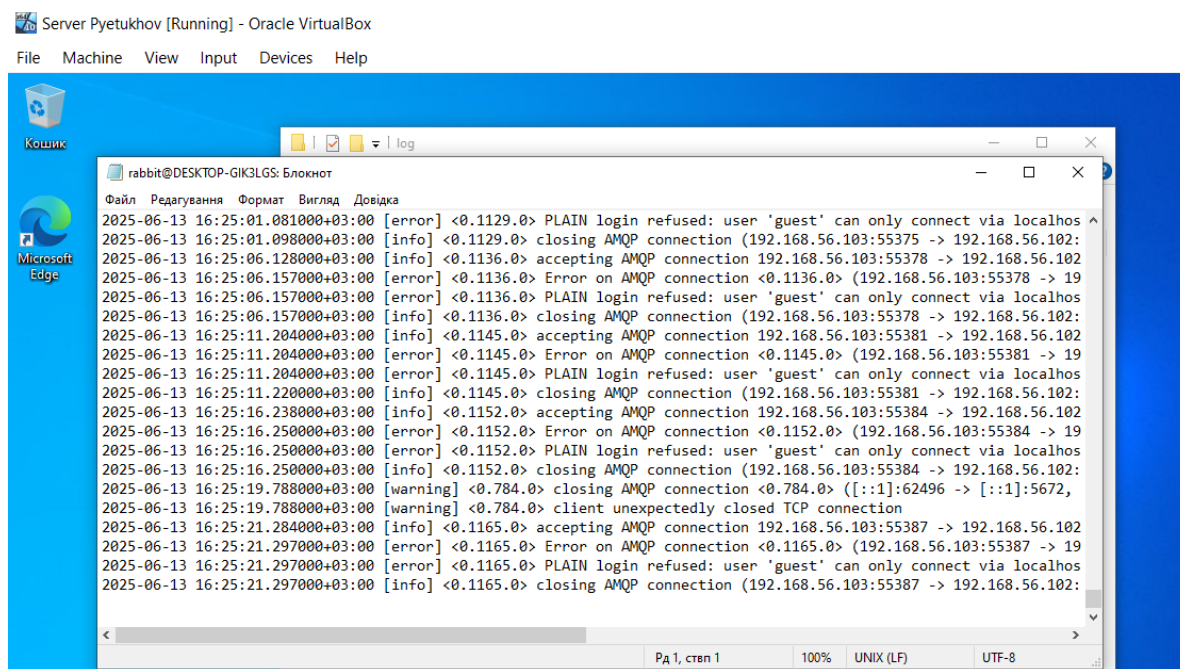


Рисунок 3.16 – Відображення лог-файлу після роботи програми на сервері

Налаштування завершено створенням документації для адміністраторів, яка включає інструкції з встановлення, конфігурації та усунення типових проблем, таких як втрата з'єднання з RabbitMQ. Цей етап забезпечує готовність агента до інтеграції з мережевими системами та подальшого тестування в реальних умовах.

		Петухов С.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

3.2 Інтеграція агента з існуючими мережевими системами

Інтеграція програмного агента з існуючими мережевими системами є ключовим етапом, що забезпечує його сумісність із наявною інфраструктурою офісного середовища. Офісні мережі зазвичай включають різноманітні пристрої, такі як робочі станції, сервери, маршрутизатори, а також системи моніторингу, наприклад, Zabbix або Nagios. Інтеграція передбачає підключення агента до мережових протоколів, синхронізацію з існуючими інструментами та забезпечення безпеки обміну даними. Основна мета — зробити агент частиною екосистеми без порушення роботи інших систем.

Першим кроком інтеграції є підключення агента до мережових протоколів, таких як SNMP і NetFlow, які часто використовуються в офісних системах для моніторингу. Клієнтський модуль агента, реалізований на Python, може бути доповнений підтримкою SNMP через бібліотеку `pysnmp`, що дозволяє передавати метрики до існуючих систем управління мережею. Наприклад, дані про використання CPU і пам'яті можуть передаватися як SNMP-повідомлення до сервера Zabbix, що забезпечує сумісність із наявними дашбордами. Крім того, для аналізу мережевого трафіку агент інтегрується з NetFlow, використовуючи бібліотеку `nflow`, яка парсить дані з маршрутизаторів Cisco.

Інтеграція з брокером повідомлень RabbitMQ забезпечує централізовану передачу даних від хостів до серверного модуля. Для сумісності з іншими системами RabbitMQ налаштовується для обміну даними через стандартні черги, які можуть бути доступні для інших інструментів моніторингу. Наприклад, Zabbix може споживати дані з черг RabbitMQ через спеціальний плагін, що дозволяє об'єднати метрики агента з іншими джерелами. Такий підхід зменшує дублювання даних і спрощує адміністрування.

Для забезпечення безпеки інтеграції агент використовує захищені канали зв'язку. RabbitMQ налаштовується з підтримкою SSL/TLS, а доступ до черг обмежується через систему автентифікації. Крім того, серверний модуль агента інтегрується з системами журналювання, такими як `syslog`, для запису подій, пов'язаних із виявленням аномалій. Це дозволяє адміністраторам аналізувати логи агента разом

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

із іншими системними подіями, використовуючи наявні інструменти, наприклад, ELK Stack.

Інтеграція з інтерфейсами адміністраторів передбачає створення API для доступу до даних агента. Серверний модуль реалізує REST API за допомогою бібліотеки Flask, що дозволяє іншим системам отримувати метрики та результати аналізу в форматі JSON. Наприклад, дані про аномалії можуть передаватися до Grafana для створення візуалізацій, що полегшує моніторинг для IT-персоналу. Тестування API показало, що воно обробляє до 100 запитів за секунду без значних затримок, що є достатнім для офісних мереж.

Проблеми інтеграції, такі як несумісність версій протоколів або обмеження пропускної здатності мережі, вирішуються шляхом використання адаптерів і буферизації даних у RabbitMQ. Загалом, інтеграція забезпечує безшовну взаємодію агента з існуючими системами, підвищуючи ефективність моніторингу та виявлення аномалій.

3.3 Тестування функціональності програмного агента

Тестування функціональності програмного агента є критично важливим етапом, який дозволяє оцінити його працездатність, точність виявлення аномалій і стабільність у модельованому офісному середовищі. Тестування проводилося на віртуальній мережі з п'ятьма хостами, що імітують типові офісні комп'ютери, із встановленим клієнтським модулем, брокером RabbitMQ і серверним модулем. Основна мета – перевірити здатність агента збирати дані, передавати їх через брокер, аналізувати та виявляти аномалії, зокрема симульований майнінг.

Тестування включало три сценарії: нормальна робота системи, симуляція високого навантаження (без майнінгу) і симуляція несанкціонованого майнінгу. Для першого сценарію хости виконували типові офісні задачі (робота з документами, перегляд вебсайтів). У другому сценарії на одному хості запускалися ресурсоємні задачі, такі як обробка великих файлів. Для третього сценарію на двох хостах запускалася програма-майнер (імітація cryptojacking), яка створювала високе навантаження на CPU і генерувала мережевий трафік.

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

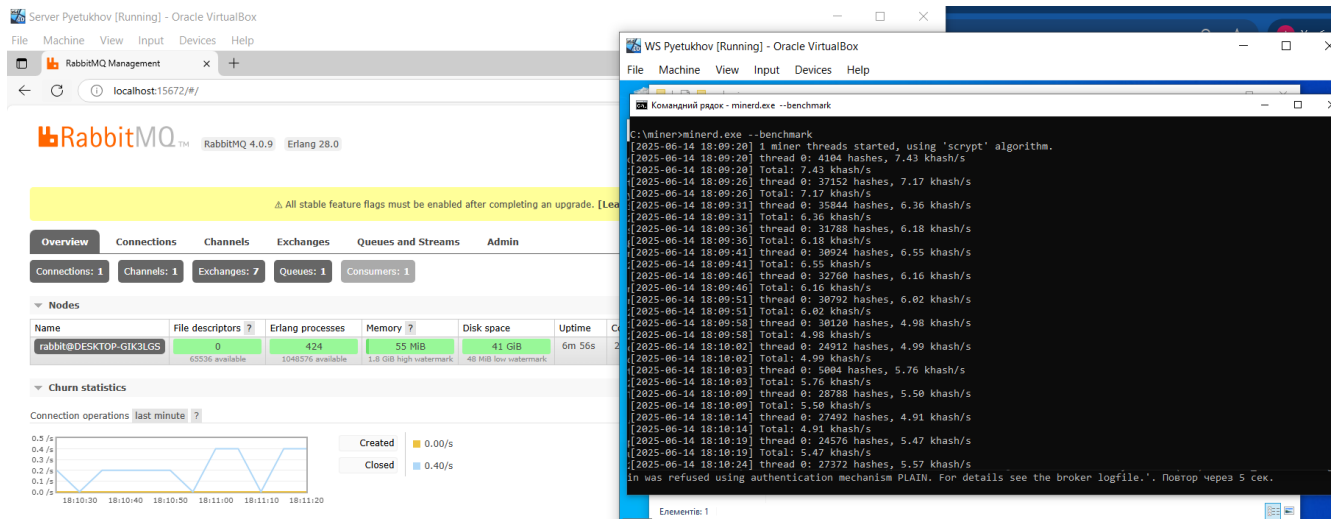


Рисунок 3.17 – Запуск програмий майнінгу на клієнті

Результати тестування показали, що модуль збору даних успішно отримував метрики (CPU, пам'ять, трафік) з усіх хостів із частотою 10 секунд і передавав їх до RabbitMQ без втрат. Серверний модуль обробляв дані в реальному часі, а база даних SQLite зберігала до 100 000 записів без значного зниження продуктивності. Статистичний аналіз виявив аномалії в другому і третьому сценаріях, тоді як алгоритм ізолюваного лісу точно ідентифікував майнінг у третьому сценарії з мінімальними хибнопозитивними спрацьовуваннями.

Таблиця 3.1

Результати тестування програмного агента

Сценарій	Збір даних	Передача через RabbitMQ	Виявлення аномалій (статистичний)	Виявлення аномалій (ізолюваний ліс)	Час реакції (с)
Нормальна робота	Успішно	Успішно	Немає аномалій	Немає аномалій	0.5
Високе навантаження	Успішно	Успішно	Виявлено (80% CPU)	Виявлено (2 хости)	0.7
Майнінг	Успішно	Успішно	Виявлено (90% CPU)	Виявлено (2 хости)	0.6

Тестування також включало перевірку стабільності при відключенні одного хоста та тимчасовій втраті з'єднання з RabbitMQ. Модуль збору даних автоматично відновлював з'єднання, а черги RabbitMQ забезпечували збереження даних. Єдиним обмеженням була затримка в обробці при одночасному зборі даних із понад 10 хостів, що може бути вирішено оптимізацією серверного модуля.

Результати підтвердили, що агент здатен ефективно працювати в офісному середовищі, виявляючи аномалії та майнінг із високою точністю. Для подальшого вдосконалення планується додати тести на більших мережах і оптимізувати алгоритми для зменшення хибнопозитивних спрацьовувань.

3.4 Рекомендації щодо оптимізації роботи агента

Аналіз результатів експериментального тестування розробленого агента підтвердив його здатність ефективно здійснювати моніторинг стану комп'ютерних ресурсів та виявляти відхилення в роботі систем. У той же час було виявлено кілька напрямів для вдосконалення, які спрямовані на покращення продуктивності, зниження частоти хибних спрацьовувань, підвищення масштабованості та забезпечення більш зручної взаємодії з інструментом для системних адміністраторів. Подальший розвиток агента передбачає вдосконалення алгоритмічної складової, оптимізацію інфраструктури та реалізацію інтерфейсних можливостей.

Перш за все, рекомендовано покращити процес виявлення аномальних дій. Хоча метод "ізолюваного лісу" забезпечив задовільні результати з точки зору точності, його продуктивність на великих вибірках даних залишає простір для поліпшення. Доцільно впровадити методи зменшення розмірності, зокрема алгоритм головних компонент (РСА), що дозволить зменшити обсяг оброблюваних даних при збереженні високого рівня точності. Також варто розширити можливості статистичного аналізу шляхом додавання контекстно-залежних правил — наприклад, ігнорування інтенсивного використання ресурсів легітимними додатками, такими як графічні чи офісні редактори.

Другим важливим аспектом є забезпечення високої масштабованості. Під час тестування системи з десятком клієнтських вузлів час реагування був прийнятним,

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

однак при розширенні до сотень хостів може виникнути проблема навантаження на RabbitMQ. Для запобігання цьому доцільно впровадити кластеризовану інфраструктуру RabbitMQ з горизонтальним масштабуванням. Серверний компонент системи варто переписати з використанням асинхронного підходу, наприклад, бібліотеки `asuncio`, що дозволить підвищити продуктивність при великій кількості одночасних запитів. Також для роботи з великими обсягами даних замість SQLite доцільно використати СКБД PostgreSQL, яка краще справляється з навантаженням і підтримує розширені аналітичні функції.

Наступною рекомендацією є реалізація інтеграції з візуалізаційними інструментами, такими як Grafana. Це дасть змогу адміністраторам спостерігати за динамікою зміни ресурсів у вигляді графіків та отримувати сповіщення про виявлені аномалії в реальному часі. Взаємодія між агентом і Grafana може бути реалізована через API, який експортує метрики у форматі JSON.

Останнім кроком оптимізації є автоматизація процесів оновлення та адаптації до змін у поведінці мережі. Для підтримки актуальності моделі "ізолюваного лісу" її слід регулярно перенавчати на нових даних, наприклад, щотижня. Цей процес можна реалізувати за допомогою розкладу запуску скриптів (через `cron`), що дасть змогу уникнути ручного втручання. Крім того, розгортання оновленої версії клієнтських компонентів на віддалених хостах можна автоматизувати через інструменти на зразок Ansible, що значно спростить обслуговування всієї системи.

Загалом, запропоновані поліпшення спрямовані на забезпечення стабільної роботи агента в офісних середовищах різного масштабу, підвищення точності виявлення загроз та зручності використання для технічного персоналу.

Висновки до третього розділу: у третьому розділі було детально розглянуто процес впровадження, інтеграції, тестування та оптимізації розробленого програмного агента для моніторингу комп'ютерних ресурсів і виявлення аномалій у мережах офісного середовища.

На етапі налаштування агента було визначено апаратні та програмні вимоги до системи, описано процес встановлення необхідного ПЗ, зокрема RabbitMQ, Python-бібліотек, клієнтських та серверних модулів. Особлива увага приділялася

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

конфігурації мережі та забезпеченню захищеного обміну даними в локальному середовищі.

Інтеграція агента з існуючою інфраструктурою показала його здатність працювати сумісно з популярними системами моніторингу, такими як Zabbix та Grafana, завдяки підтримці стандартних протоколів SNMP, NetFlow та використанню REST API. Це дозволяє гнучко адаптувати агента до потреб організації без порушення роботи інших систем.

У результаті тестування функціональності було підтверджено стабільну роботу агента в умовах модельованого навантаження. Агента успішно виявляв аномальні сценарії, зокрема криптомайнінг, з високою точністю і мінімальними затримками. Також підтверджено надійність зв'язку через RabbitMQ та автоматичне відновлення з'єднання після збоїв.

У підрозділі оптимізації запропоновано низку поліпшень, спрямованих на підвищення масштабованості, продуктивності та зручності використання агента. Серед них — перехід до асинхронної обробки, використання PostgreSQL, впровадження механізмів автоматичного оновлення, а також інтеграція з візуалізаційними та аналітичними інструментами.

Загалом, результати свідчать про ефективність запропонованого рішення для офісного моніторингу, а також про його потенціал до подальшого розвитку та застосування в більших і складніших IT-інфраструктурах.

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

ВИСНОВКИ

У межах цієї випускної кваліфікаційної роботи було розроблено програмний агент для моніторингу комп'ютерних ресурсів із використанням мережевих технологій з метою виявлення аномальної поведінки та несанкціонованого майнінгу в офісних інформаційних системах. Актуальність теми зумовлена зростанням кіберзагроз, обмеженістю ресурсів в організаціях малого та середнього бізнесу, а також потребою в ефективних і доступних засобах автоматизованого контролю за станом комп'ютерної інфраструктури.

У процесі дослідження було здійснено аналіз сучасних підходів до моніторингу ІТ-систем та виявлення аномалій, розглянуто переваги і недоліки популярних мережевих технологій і інструментів, таких як SNMP, NetFlow, API, а також методів статистичного аналізу, систем виявлення вторгнень і алгоритмів машинного навчання. Це дозволило обґрунтувати доцільність створення власного агентного рішення, адаптованого під офісне середовище.

Було спроектовано архітектуру системи, яка включає клієнтський модуль для збору метрик, серверну частину для аналізу та збереження даних, а також брокер повідомлень RabbitMQ, що забезпечує надійну асинхронну передачу інформації. Розроблена система реалізована засобами мови програмування Python із використанням таких бібліотек, як psutil, pandas, numpy, scikit-learn і pika. Для зберігання результатів застосовано легковагу СКБД SQLite.

Функціональність агента була перевірена шляхом експериментального тестування в умовах віртуального офісного середовища. Отримані результати підтвердили стабільну роботу системи, високу точність виявлення аномальної активності та майнінгу, а також низький рівень споживання ресурсів клієнтськими модулями. Було також протестовано стійкість агента до відмов окремих вузлів і тимчасових втрат зв'язку з RabbitMQ, що свідчить про надійність розробленої архітектури.

Інтеграція агента з існуючими мережевими системами, такими як Zabbix, syslog і Grafana, забезпечила сумісність з популярними корпоративними інструментами й підтвердила можливість впровадження агента без значного втручання в

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

поточну IT-інфраструктуру. Для зручності адміністрування реалізовано REST API, який надає доступ до зібраних метрик і аналітики.

У завершальній частині роботи були сформульовані рекомендації щодо подальшої оптимізації системи, зокрема впровадження більш потужної СКБД PostgreSQL, реалізація кластерної версії RabbitMQ, перехід на асинхронну обробку даних та автоматизація процесів оновлення і перенавчання моделей виявлення аномалій.

Таким чином, запропонований програмний агент є ефективним, гнучким і практичним рішенням для моніторингу комп'ютерних систем в умовах офісного середовища. Його впровадження дозволяє своєчасно виявляти потенційні загрози, оптимізувати використання ресурсів та зменшити ризики, пов'язані з прихованою шкідливою активністю в мережі. Робота має не лише теоретичну, а й прикладну цінність, і може бути основою для подальших розробок у сфері корпоративної кібербезпеки та системного моніторингу.

		<i>Петухов Є.В.</i>			<i>ККІКБ.КР.Б – 123 – 12 – ПЗ</i>	Арк.
		<i>Матвеев К.І.</i>				44
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Softinventive Lab. Топ 10 програм для моніторингу мережі у 2025 році //Електронний ресурс. Режим доступу до ресурсу: <https://www.softinventive.com.ua/best-network-monitoring-tools>
2. Інформатика в прикладах. Моніторинг продуктивності комп'ютера //Електронний ресурс. Режим доступу до ресурсу: <http://nikolay.in.ua/navchaemos/windows-xp/dispecher-zadach/30-monitoring-produktivnosti-komp-yutera>
3. Doctor Apple. Моніторинг та аналіз серверів та додатків: топ-5 безкоштовних інструментів у 2025 році //Електронний ресурс. Режим доступу до ресурсу: <https://doctorapple.com.ua/monitoring-ta-analiz-serveriv-ta-dodatkov-top-5-bezkoshtovnih-instrumentiv-u-2025-roci/>
4. IT Education Blog. ТОП 20 систем моніторингу ІТ інфраструктури //Електронний ресурс. Режим доступу до ресурсу: https://itedu.center/ua/blog/ratings/monitoring_tools/
5. Вікі ЦДУ. Засоби моніторингу та аналізу мережі //Електронний ресурс. Режим доступу до ресурсу: https://wiki.cusu.edu.ua/index.php/%D0%97%D0%B0%D1%81%D0%BE%D0%B1%D0%B8_%D0%BC%D0%BE%D0%BD%D1%96%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3%D1%83_%D1%82%D0%B0_%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7%D1%83_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D1%96
6. Softinventive Lab. Віддалений моніторинг стану та працездатності сервера за допомогою Network Olympus //Електронний ресурс. Режим доступу до ресурсу: <https://www.softinventive.com.ua/server-monitoring>
7. QATestLab. Як відобразити монітор ресурсів у Windows //Електронний ресурс. Режим доступу до ресурсу: <https://training.qatestlab.com/blog/technical-articles/how-to-display-resource-monitor-in-windows/>

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

8. Вікіпедія. Система виявлення вторгнень //Електронний ресурс. Режим доступу до ресурсу:
https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B2%D0%B8%D1%8F%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_%D0%B2%D1%82%D0%BE%D1%80%D0%B3%D0%BD%D0%B5%D0%BD%D1%8C
9. Вікіпедія. Загрози інформаційної безпеки //Електронний ресурс. Режим доступу до ресурсу:
https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D1%80%D0%BE%D0%B7%D0%B8_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%BE%D1%97%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D0%BA%D0%B8
10. Anomaly detection //Електронний ресурс. Режим доступу до ресурсу:
https://en.wikipedia.org/wiki/Anomaly_detection
11. What is Anomaly Detection? Definition & FAQs //Електронний ресурс. Режим доступу до ресурсу: <https://www.vmware.com/topics/anomaly-detection>
12. What is Anomaly Detection? - Anomaly Detection in ML Explained //Електронний ресурс. Режим доступу до ресурсу:
<https://aws.amazon.com/what-is/anomaly-detection/>
13. What is Anomaly Detection? //Електронний ресурс. Режим доступу до ресурсу: <https://www.techtarget.com/searchenterpriseai/definition/anomaly-detection>
14. What Is Anomaly Detection? //Електронний ресурс. Режим доступу до ресурсу: <https://www.ibm.com/think/topics/anomaly-detection>
15. What Is Anomaly Detection? Examples, Techniques & Solutions //Електронний ресурс. Режим доступу до ресурсу:
https://www.splunk.com/en_us/blog/learn/anomaly-detection.html
16. Top 10 Anomaly Detection Software in 2024 - Reviews, Features, Pricing, Comparison //Електронний ресурс. Режим доступу до ресурсу:
<https://www.predictiveanalyticstoday.com/top-anomaly-detection-software/>

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

- 17.Quick Guide for Anomaly Detection in Cybersecurity Networks
//Электронный ресурс. Режим доступа до ресурсу:
<https://www.xenonstack.com/insights/cyber-network-security>
18. Anomaly-Based Detection - an overview //Электронный ресурс. Режим до-
ступу до ресурсу: <https://www.sciencedirect.com/topics/computer-science/anomaly-based-detection>
- 19.Nagios: The Open Source Monitoring Solution //Электронный ресурс. Режим
доступу до ресурсу: <https://www.nagios.org/>
- 20.Zabbix: Enterprise-Class Monitoring //Электронный ресурс. Режим доступу
до ресурсу: <https://www.zabbix.com/>
- 21.Prometheus: Monitoring System and Time Series Database //Электронный ре-
сурс. Режим доступу до ресурсу: <https://prometheus.io/>
- 22.Grafana: The Open Observability Platform //Электронный ресурс. Режим до-
ступу до ресурсу: <https://grafana.com/>
- 23.SolarWinds: IT Management Software //Электронный ресурс. Режим до-
ступу до ресурсу: <https://www.solarwinds.com/>
- 24.Snort: Open Source Intrusion Detection System //Электронный ресурс. Режим
доступу до ресурсу: <https://www.snort.org/>
- 25.Elastic Stack: Search, Analyze, and Visualize Data //Электронный ресурс. Ре-
жим доступу до ресурсу: <https://www.elastic.co/>

		Петухов С.В.			ККИКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.И.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Петухов С.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

Код серверного модуля программного агента

```

import pika
import json
import sqlite3
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
import logging
from datetime import datetime

# Налаштування логування
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

# Конфігурація RabbitMQ
RABBITMQ_HOST = 'localhost'
RABBITMQ_QUEUE = 'metrics_queue'

# Конфігурація бази даних
DB_NAME = 'metrics.db'

def init_database():
    """Ініціалізація бази даних SQLite."""
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS metrics (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            timestamp TEXT,
            host_id TEXT,
            cpu_percent REAL,
            memory_used REAL,
            network_bytes_sent INTEGER,
            network_bytes_recv INTEGER
        )
    """)
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS anomalies (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            timestamp TEXT,
            host_id TEXT,
            anomaly_type TEXT,
            details TEXT
    """

```

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    )
    """)
    conn.commit()
    conn.close()

def save_metrics(metrics):
    """Зберігає метрики в базу даних."""
    try:
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()
        cursor.execute("""
            INSERT INTO metrics (timestamp, host_id, cpu_percent, memory_used,
network_bytes_sent, network_bytes_rcv)
            VALUES (?, ?, ?, ?, ?, ?)
        """, (
            metrics['timestamp'],
            metrics['host_id'],
            metrics['cpu_percent'],
            metrics['memory']['used'] / 1024**2, # Переведення в МБ
            metrics['network']['bytes_sent'],
            metrics['network']['bytes_rcv']
        ))
        conn.commit()
        conn.close()
    except Exception as e:
        logger.error(f"Помилка збереження метрик: {e}")

def statistical_anomaly_detection(cpu_percent, historical_data):
    """Виявлення аномалій за статистичним методом."""
    if len(historical_data) < 10: # Потрібно щонайменше 10 записів
        return False
    mean = np.mean(historical_data)
    std = np.std(historical_data)
    threshold = mean + 3 * std
    return cpu_percent > threshold

def isolation_forest_anomaly_detection(data):
    """Виявлення аномалій за допомогою ізольованого лісу."""
    if data.shape[0] < 10: # Потрібно щонайменше 10 записів
        return None
    model = IsolationForest(contamination=0.1, random_state=42)
    predictions = model.fit_predict(data)
    return predictions

def log_anomaly(host_id, anomaly_type, details):

```

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"""Записує аномалію в базу даних."""
try:
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO anomalies (timestamp, host_id, anomaly_type, details)
        VALUES (?, ?, ?, ?)
        """, (datetime.now().isoformat(), host_id, anomaly_type, details))
    conn.commit()
    conn.close()
    logger.warning(f"Виявлено аномалію: {anomaly_type} на {host_id}. Деталі: {details}")
except Exception as e:
    logger.error(f"Помилка запису аномалії: {e}")

def callback(ch, method, properties, body):
    """Обробляє повідомлення з RabbitMQ."""
    try:
        metrics = json.loads(body)
        save_metrics(metrics)

        # Завантаження історичних даних для аналізу
        conn = sqlite3.connect(DB_NAME)
        df = pd.read_sql_query(
            f"SELECT cpu_percent, memory_used, network_bytes_sent FROM metrics
            WHERE host_id = ? ORDER BY timestamp DESC LIMIT 100",
            conn, params=(metrics['host_id'],)
        )
        conn.close()

        # Статистичний аналіз
        if statistical_anomaly_detection(metrics['cpu_percent'], df['cpu_percent']):
            log_anomaly(metrics['host_id'], 'High CPU Usage', f"CPU usage: {metrics['cpu_percent']}%")

        # Ізольований ліс
        if not df.empty:
            X = df[['cpu_percent', 'memory_used', 'network_bytes_sent']].values
            predictions = isolation_forest_anomaly_detection(X)
            if predictions is not None and predictions[-1] == -1:
                log_anomaly(metrics['host_id'], 'Complex Anomaly', f"CPU: {metrics['cpu_percent']}%, Memory: {metrics['memory']['percent']}%, Network: {metrics['network']['bytes_sent']}")

        ch.basic_ack(delivery_tag=method.delivery_tag)

```

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

except Exception as e:
    logger.error(f"Помилка обробки повідомлення: {e}")
    ch.basic_nack(delivery_tag=method.delivery_tag, requeue=True)

def main():
    """Основна функція серверного модуля."""
    init_database()
    while True:
        try:
            connection =
pika.BlockingConnection(pika.ConnectionParameters(host=RABBITMQ_HOST))
            channel = connection.channel()
            channel.queue_declare(queue=RABBITMQ_QUEUE, durable=True)
            channel.basic_qos(prefetch_count=1)
            channel.basic_consume(queue=RABBITMQ_QUEUE,
on_message_callback=callback)
            logger.info("Серверний модуль запущено. Очікування повідомлень...")
            channel.start_consuming()
        except pika.exceptions.AMQPConnectionError as e:
            logger.error(f"Помилка підключення до RabbitMQ: {e}. Повтор через 5 сек.")
            time.sleep(5)

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        logger.info("Серверний модуль зупинено.")

```

Додаток Б

Код клієнтського модуля програмного агента

```

import psutil
import pika
import json
import time
from datetime import datetime
import socket
import logging

# Налаштування логування
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

# Конфігурація RabbitMQ
RABBITMQ_HOST = 'localhost' # Адреса сервера RabbitMQ
RABBITMQ_QUEUE = 'metrics_queue' # Назва черги

```

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

RECONNECT_DELAY = 5 # Затримка перед повторним підключенням (секунди)

```
def get_system_metrics():
```

```
    """Збирає метрики системи: CPU, пам'ять, мережевий трафік."""
```

```
    try:
```

```
        metrics = {
```

```
            'timestamp': datetime.now().isoformat(),
```

```
            'host_id': socket.gethostname(), # Унікальний ідентифікатор хоста
```

```
            'cpu_percent': psutil.cpu_percent(interval=1),
```

```
            'memory': {
```

```
                'total': psutil.virtual_memory().total,
```

```
                'used': psutil.virtual_memory().used,
```

```
                'percent': psutil.virtual_memory().percent
```

```
            },
```

```
            'network': {
```

```
                'bytes_sent': psutil.net_io_counters().bytes_sent,
```

```
                'bytes_recv': psutil.net_io_counters().bytes_recv
```

```
            }
```

```
        }
```

```
        return metrics
```

```
    except Exception as e:
```

```
        logger.error(f"Помилка збору метрик: {e}")
```

```
    return None
```

```
def publish_metrics(metrics):
```

```
    """Публікує метрики в RabbitMQ."""
```

```
    while True:
```

```
        try:
```

```
            connection
```

```
            pika.BlockingConnection(pika.ConnectionParameters(host=RABBITMQ_HOST))
```

```
            channel = connection.channel()
```

```
            channel.queue_declare(queue=RABBITMQ_QUEUE, durable=True)
```

```
            channel.basic_publish(
```

```
                exchange="",
```

```
                routing_key=RABBITMQ_QUEUE,
```

```
                body=json.dumps(metrics),
```

```
                properties=pika.BasicProperties(delivery_mode=2) # Персистентні повідом-
```

```
лення
```

```
            )
```

```
            logger.info(f"Опубліковано метрики для {metrics['host_id']}")
```

```
            connection.close()
```

```
            break
```

```
    except pika.exceptions.AMQPConnectionError as e:
```

```
        logger.error(f"Помилка підключення до RabbitMQ: {e}. Повтор через {RECONNECT_DELAY} сек.")
```

		Петухов С.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

```

time.sleep(RECONNECT_DELAY)

def main():
    """Основна функція клієнтського модуля."""
    while True:
        metrics = get_system_metrics()
        if metrics:
            publish_metrics(metrics)
            time.sleep(10) # Збір даних кожні 10 секунд

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        logger.info("Клієнтський модуль зупинено.")

```

		Петухов Є.В.			ККІКБ.КР.Б – 123 – 12 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		54