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
“____ ” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації”

Виконав здобувач вищої освіти
IV курсу, групи КБ-22МБ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Меркулов В.Р.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“* Кібербезпека *”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Меркулова Владислава Руслановича
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації*

2. Керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від 17.01.2025 року № 51-02

3. Строк подання роботи до захисту *24.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію

6. Висновки.

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

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

7. Дата видачі завдання « 17 » січня 2025 р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем керування	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	15.05.2025 р.	
8.	Попередній захист роботи	24.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Дрєєв О.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Меркулов В.Р.
(прізвище та ініціали)

АНОТАЦІЯ

Меркулов В.Р. Програмне забезпечення системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації.

Метою розробки є програмне забезпечення системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації.

Результат роботи – програмна реалізація системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації.

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 3.10.

Ключові слова: кібербезпека, моніторинг серверів, перерозподіл ресурсів, цілісність

ABSTRACT

Merkulov V.R. Cybersecurity system software monitoring servers for the purpose of timely redistribution of resources to ensure information integrity. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor) level of higher education, software has been developed that is intended for cybersecurity system monitoring servers for the purpose of timely redistribution of resources to ensure information integrity.

The purpose of the development is the cybersecurity system software monitoring servers for the purpose of timely redistribution of resources to ensure information integrity.

The result of work is a software implementation of the cybersecurity system monitoring servers for the purpose of timely redistribution of resources to ensure information integrity.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A user-friendly interface has been developed. Instructions for working with the software tools have been provided.

The program can be used on an PC with Windows 10/11 OS.

The program was developed in the Python 3.10 environment.

Keywords: cybersecurity, server monitoring, resource redistribution, integrity

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

JWT – JSON Web Token

ПЗ – програмне забезпечення

ІТ – інформаційні технології

HTTP – Hypertext Transfer Protocol

PromQL – Prometheus Query Language

API – Application Programming Interface

ELK – Elasticsearch, Logstash, Kibana

PRTG – Paessler Router Traffic Grapher

DevOps – Development Operations

SNMP – Simple Network Management Protocol

OpenNMS – Open Network Monitoring System

APM – Application Performance Management

BMC – Business Management Corporation

JSON – JavaScript Object Notation

SMTP – Simple Mail Transfer Protocol

CPU – Central processing unit

DDoS – Distributed Denial of Service

ОС – операційна система

DES – Data Encryption Standard

IP – Internet Protocol

FMS – Flexible Monitoring System

SMS – Short Message Service

HTML – HyperText Markup Language

Gmail – Google Mail

ВСТУП

З розвитком цифрових технологій, обсяги інформації та навантаження на серверні системи зростають. Порушення цілісності або втрата даних через перевантаження серверів може призвести до значних втрат для різних підприємств [1].

Для України, яка активно оновлює державні та комерційні структури, особливо важливо впроваджувати надійні та безпечні системи автоматизованого моніторингу серверної частини. Сучасні інструменти часто не дозволяють реагувати на зміну навантаження або є складними для інтеграції у організації та підприємства. Таким чином, створення ефективного програмного забезпечення для моніторингу серверів із функцією перерозподілу ресурсів є досить актуальним [2].

Метою роботи є розробка програмного забезпечення системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації. Система повинна виявляти перевантаження, здійснювати дії з розвантаження серверів, логувати події, інформувати про критичні ситуації, а також забезпечувати зручний доступ до налаштувань і відповідних результатів моніторингу.

Для досягнення мети були поставлені наступні завдання:

- реалізувати модуль моніторингу серверів для збору метрик;
- створити модуль аналізу отриманих метрик;
- розробити модуль перерозподілу ресурсів;
- впровадити модуль оповіщення через електронну пошту;
- забезпечити логування подій;
- створити модуль конфігурації;
- реалізувати модуль безпеки;
- провести тестування системи;

– розробити документацію.

Розроблене програмне забезпечення може бути перенесене у різні структури та сфери використання. Система забезпечує виявлення перевантажень, оперативне реагування, а також перегляд відповідної інформації у вигляді логів. Вона є адаптивною до зміни, не вимагає спеціального обладнання і може бути використана у середовищах з обмеженими ресурсами. Отримані результати можуть слугувати для подальшої розробки більш складних систем моніторингу.

Автоматизоване управління ресурсами серверів стає все більш необхідним для підприємств. Реалізація програмного забезпечення, яке поєднує гнучкість до змін, зручність та безпеку, зумовлює підвищення ефективності для різної інфраструктури.

Окрім основної функціональності, система враховує вимоги до безпеки, підтримуючи різноманітні механізми захисту від потенційних кібератак. За рахунок поділу системи на окремі частини реалізовано масштабування функціоналу, що дозволяє досить легко адаптувати програму до потреб підприємства або інфраструктури. Поєднання інструментів тестування, конфігурації та інтерфейсу підвищує зручність використання системи, що робить її більш ефективним інструментом для контролю та підтримки даного серверного середовища.

Виконана робота не лише демонструє технічну реалізацію системи з ступенем автономності, безпеки та адаптивності, а й підтверджує доцільність її використання для підвищення ефективності керування серверними ресурсами в умовах сучасних викликів інформаційної безпеки.

Таким чином, розроблене програмне забезпечення може бути інтегроване у різні сфери використання. Система забезпечує виявлення перевантажень, перерозподіл ресурсів, надсилання сповіщень, оновлення конфігурації, моніторинг, аналіз та захист даних. Вона є адаптивною до змін та придатна для подальшого вдосконалення.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система, що розробляється в межах даної роботи, є програмним забезпеченням системи кібербезпеки для моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації. Основне призначення системи – забезпечити надійне виявлення критичних станів серверної системи, попередити перевантаження та реалізувати механізми реакції, які дозволяють уникнути простоїв, втрати даних та порушення роботи відповідних сервісів.

Система виконує кілька ключових функцій:

1) Моніторинг – отримання даних про стан серверів, таких як навантаження на процесор, використання оперативної пам’яті, заповнення дискового простору, обсяг переданої/отриманої мережевої інформації.

2) Аналіз даних – обробка зібраних метрик із метою виявлення ознак критичних навантажень або потенційно небезпечних станів системи. На основі заданих порогових значень система визначає, чи потребує ситуація втручання.

3) Розподіл ресурсів – виконання ряду дій щодо оптимізації навантаження, зокрема: очищення тимчасових файлів, спроба вивільнити ресурси, розподіл навантаження між іншими вузлами системи.

4) Система сповіщень – надсилання повідомлень за допомогою електронної пошти у разі перевищення критичних порогів або виникнення підозрілої активності.

5) Логування – запис усіх ключових подій у лог-файли, що дозволяє відслідковувати історію роботи системи, діагностувати помилки та аналізувати інциденти.

6) Безпека – забезпечення інформаційної безпеки за допомогою механізмів шифрування даних, виявлення підозрілої активності, автентифікації та авторизації користувачів через JWT токени, захисту доступу до ключових функцій системи.

7) Управління конфігурацією – можливість зміни параметрів моніторингу без перезапуску системи.

8) Тестування – забезпечення перевірки коректності роботи функціональних блоків.

9) Документація – відображення інструкції користувача для зручного використання.

Таким чином, призначення системи полягає у створенні комплексного рішення для безпечного, ефективного та зручного контролю за станом серверної інфраструктури. Система дозволяє не лише збирати інформацію про поточний стан, але й динамічно реагувати на виявлені загрози, що забезпечує високий рівень надійності, інформаційної безпеки та стабільності роботи серверних систем.

1.2 Область використання

Розроблена система має великий спектр можливого використання в різних сферах діяльності, де існує потреба в контролі та захисті відповідної інфраструктури.

Основні області застосування [3]:

1) Організації з великою IT-інфраструктурою – банки, оператори, страхові компанії, що потребують постійного моніторингу серверного обладнання, оскільки від безперебійної роботи їхніх систем залежить стабільність надання послуг, збереження критично важливих клієнтських даних, дотримання вимог кібербезпеки, а також швидке реагування на можливі кібератаки та програмні збої в системі.

2) Центри обробки даних – ключову роль відіграє надійність, продуктивність та інформаційна безпека серверних систем. Система дозволяє виявляти аномалії на ранньому етапі, оптимізувати розподіл навантаження та реагувати на інциденти.

3) Освітні та наукові установи – використовують серверні платформи для зберігання навчальних матеріалів, обчислень та даних. Забезпечення стабільного доступу до цих ресурсів є критично важливим для функціонування таких установ.

4) Муніципальні та державні структури – з огляду на необхідність доступу до державних реєстрів, баз даних населення, податкових систем, важливим є спостереження за серверною інфраструктурою та її захист від внутрішніх і зовнішніх загроз.

5) Малі та середні підприємства – не мають окремих команд інформаційної безпеки, але намагаються забезпечити стабільну роботу сервісів. Система дозволяє автоматизувати процес моніторингу, що зменшує витрати на обслуговування інфраструктури.

6) Хмарні провайдери та компанії – якість обслуговування залежить від працездатності серверів. Дана система дає змогу в режимі реального часу спостерігати за навантаженням, виявляти проблеми до їх прояву та реагувати на них.

7) Інші області, які доволі активно використовують різні серверні технології.

Таким чином, розроблена система є універсальним інструментом, який може бути впроваджений у будь-яку організацію, що має серверну інфраструктуру, потребує моніторингу, гнучкого налаштування та захисту від сучасних кібератак. Її гнучкість та простота у використанні дозволяють адаптувати рішення під конкретні потреби замовника незалежно від сфери використання [3].

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем

Аналіз ринку показує, що існує велика кількість різних інструментів для моніторингу, управління ресурсами та інформаційної безпеки [4].

Prometheus – це популярна система моніторингу з відкритим кодом, орієнтована на збирання часових рядів. Вона підтримує збирання метрик за допомогою HTTP-запитів, має власну мову запитів PromQL та активно використовується у зв'язці з Grafana для побудови графіків. Вигляд інтерфейсу Prometheus має вид [5]:

Рисунок 2.1 – Вид інтерфейсу Prometheus

Переваги [6]:

– інтуїтивно зрозумілий інтерфейс;

– відкритий код;

– працює у реальному часі.

Недоліки [6]:

– збереження даних;

– обмежений захист даних;

– налаштування.

Zabbix – система моніторингу мереж, серверів, додатків та сервісів. Підтримує збирання метрик, сповіщення, візуалізацію та інтеграцію з багатьма іншими системами. Вигляд інтерфейсу Zabbix має вид [7]:

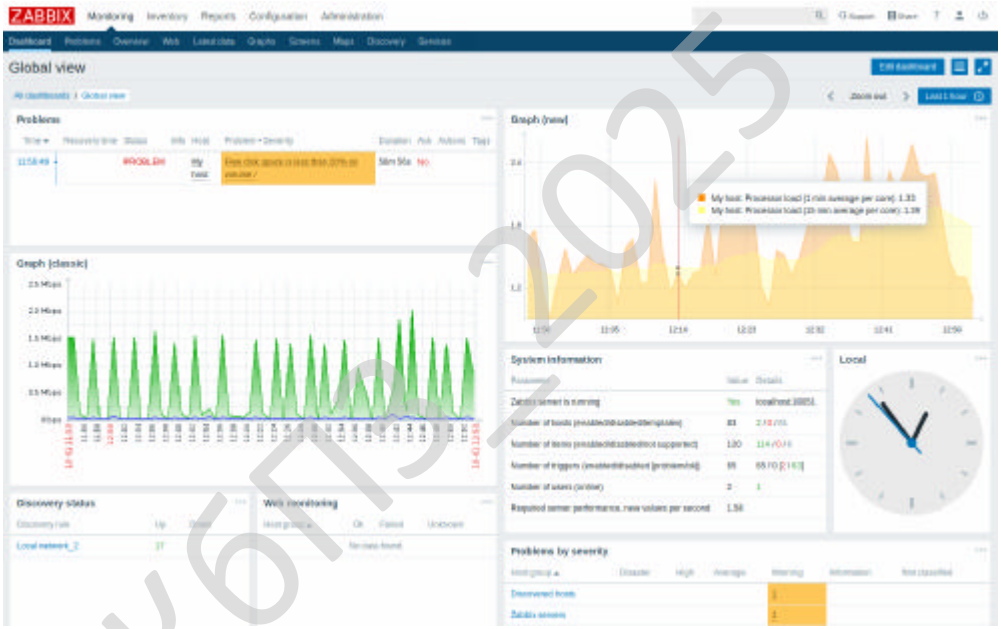

Рисунок 2.2 – Вид інтерфейсу Zabbix

Переваги [8]:

– інтуїтивно зрозумілий інтерфейс;

– масштабованість;

– ефективність.

Недоліки [9]:

– значні ресурси для запуску великих інсталяцій.

– відсутність панелей;

– додаткові налаштування.

Nagios – популярний інструмент для моніторингу серверів, сервісів та мереж. Має розгалужену систему плагінів та конфігурацій. Вигляд інтерфейсу Nagios має вид [10]:

Рисунок 2.3 – Вид інтерфейсу Nagios

Переваги [11]:

– рівень безпеки;

– масштабованість;

– створення рівнів доступу.

Недоліки [12]:

– відсутність сучасного інтерфейсу;

– обмежені можливості візуалізації;

– конфігурація.

SolarWinds – комерційна платформа моніторингу серверів та додатків, яка надає широкі можливості для підприємств. Вигляд інтерфейсу SolarWinds має вид [13]:

Рисунок 2.4 – Вид інтерфейсу SolarWinds

Переваги [13]:

– аудит;

– універсальність;

– масштабованість.

Недоліки [14]:

– налаштування;

– закритий код;

– обмежена інтеграція;

– ресурсомісткість.

Datadog – сервіс для моніторингу, аналізу метрик, додатків і логів у хмарному середовищі. Сервіс підтримує інтеграцію з понад 600 сервісами й технологіями. Завдяки функціям автоматичного виявлення аномалій, трасування запитів, побудови дашбордів і налаштування сповіщень, Datadog дозволяє своєчасно реагувати на проблеми та оптимізувати роботу систем у режимі реального часу. Вигляд інтерфейсу Datadog має вид [15]:

Рисунок 2.5 – Вид інтерфейсу Datadog

Переваги [16]:

– співпраця між командами;

– інтеграція з сервісами.

Недоліки [17]:

– індексування та зберігання логів.

Icinga – форк Nagios для моніторингу комп'ютерних систем і мереж. Вигляд інтерфейсу Icinga має вид [18]:

Рисунок 2.6 – Вид інтерфейсу Icinga

Переваги [19]:

– підтримка плагінів;

– застосування правил;

– підтримка API.

Недоліки [19]:

– автоматизація;

– сповіщення.

Netdata – система моніторингу серверів в реальному часі з відкритим кодом. Вигляд інтерфейсу Netdata має вид [20]:

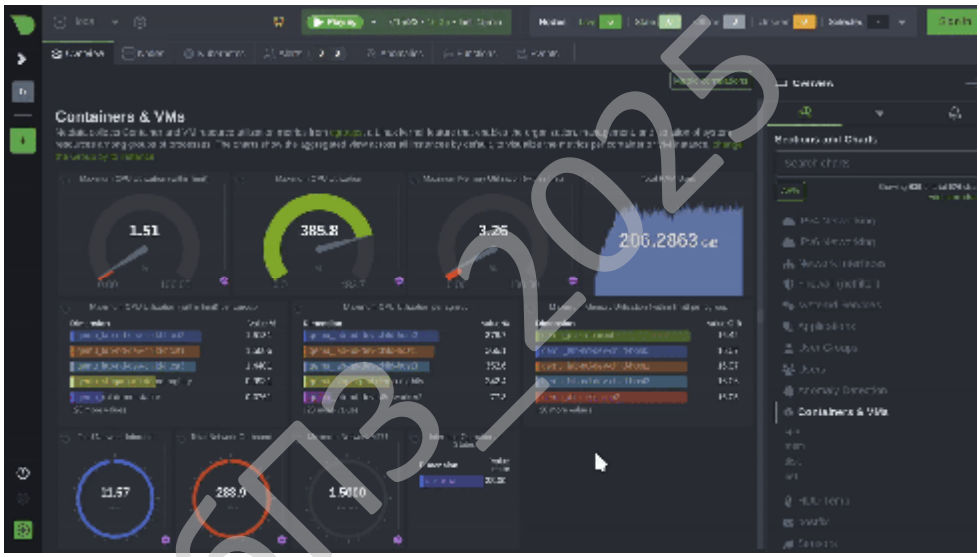

Рисунок 2.7 – Вид інтерфейсу Netdata

Переваги [21]:

– простота використання;

– параметри налаштування.

Недоліки [21]:

– сумісність;

– інтеграція з сервісами.

Checkmk – система моніторингу серверів, сховищ та баз даних. Вигляд інтерфейсу Checkmk має вид [22]:

Рисунок 2.8 – Вид інтерфейсу Checkmk

Переваги [23]:

– система сповіщень;

– масштабованість.

Недоліки [23]:

– налаштування та інтеграція з сервісами.

Splunk – платформа для збору, обробки та зберігання логів. Вигляд інтерфейсу Splunk має вид [24]:

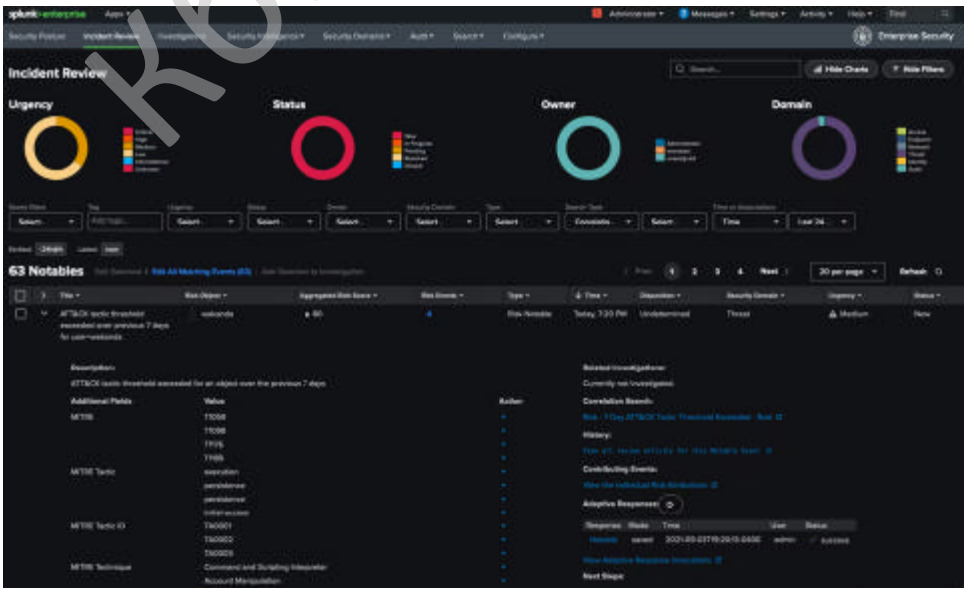

Рисунок 2.9 – Вид інтерфейсу Splunk

Переваги [25]:

– аналітика даних;

– налаштування;

– управління логами.

Недоліки [25]:

– висока вартість.

– складна архітектура.

ELK Stack – інструмент для збору, обробки та зберігання логів. Вигляд інтерфейсу ELK Stack має вид [26]:

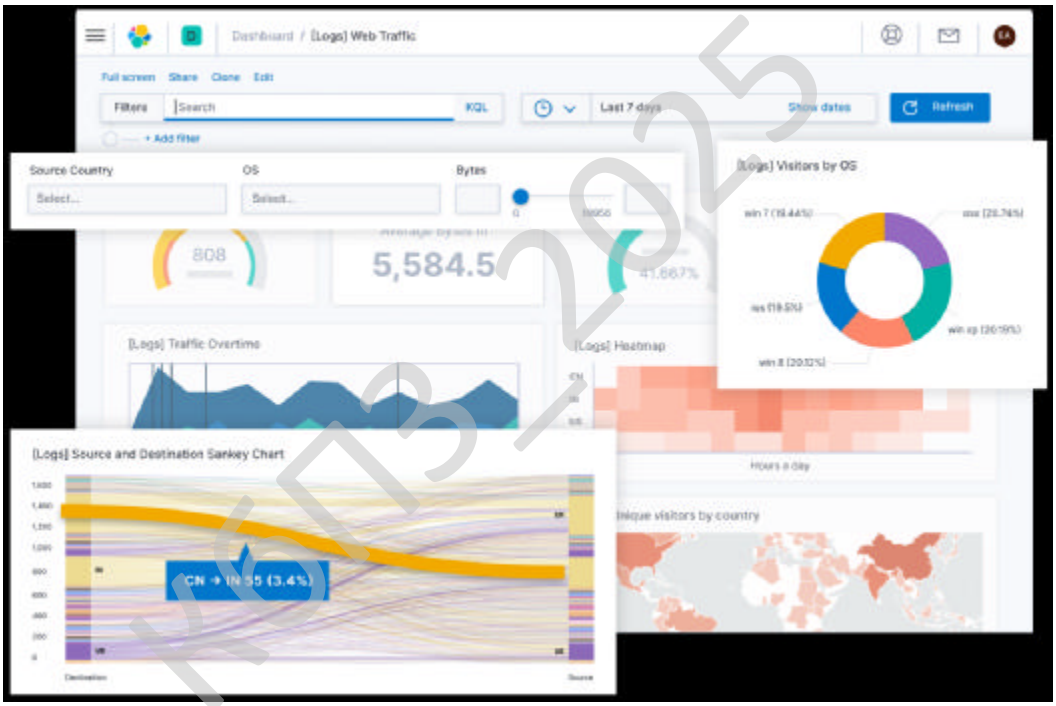

Рисунок 2.10 – Вид інтерфейсу ELK Stack

Переваги [27]:

– аналіз даних в реальному часі;

– відкритий код.

Недоліки [27]:

– складність налаштування;

– масштабування.

PRTG Network Monitor – програмне забезпечення для моніторингу мережі та серверів. Вигляд інтерфейсу PRTG Network Monitor має вид [28]:

Рисунок 2.11 – Вид інтерфейсу PRTG Network Monitor

Переваги [29]:

– моніторинг у реальному часі;

– просте налаштування.

Недоліки [29]:

– продуктивність;

– інтеграція з сервісами.

AppDynamics – платформа, яка забезпечує функціонування додатків. Вигляд інтерфейсу AppDynamics має вид [30]:

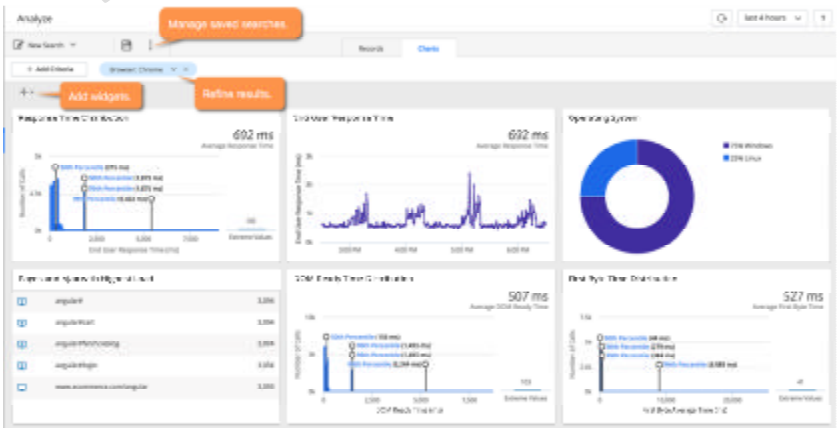

Рисунок 2.12 – Вид інтерфейсу AppDynamics

Переваги [30]:

– візуалізація;

– інтеграція з DevOps-інструментами;

– налаштування.

Недоліки [31]:

– недостатність документації;

– вартість ліцензування.

OpenNMS – платформа моніторингу та керування мережею, що підтримує SNMP, потоки даних, оповіщення та інтеграцію з іншими сервісами. Вигляд інтерфейсу OpenNMS має вид [32]:

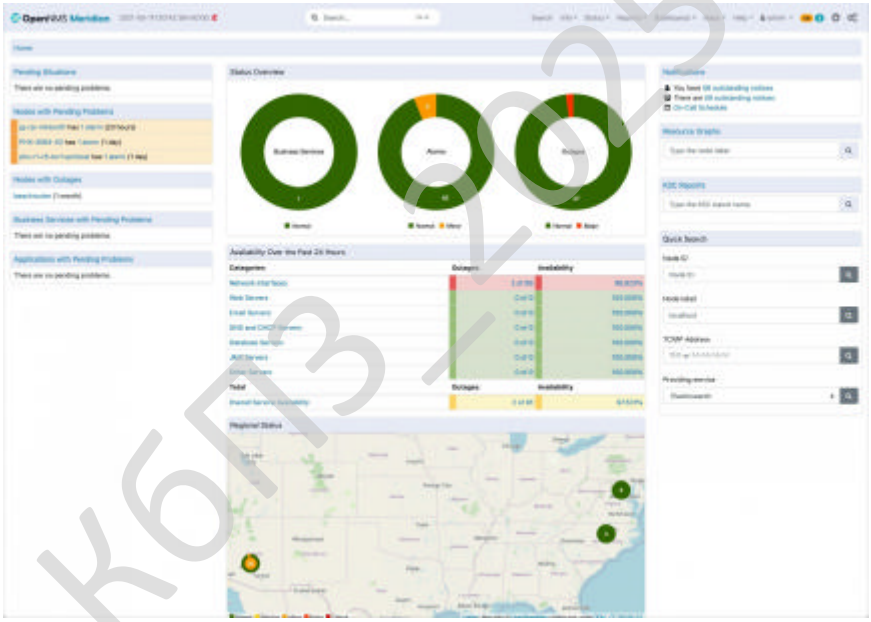

Рисунок 2.13 – Вид інтерфейсу OpenNMS

Переваги [33]:

– звітування;

– збір та аналітика даних.

Недоліки [33]:

– затримка сповіщень;

– розуміння інтерфейсу.

Site24x7 – інструмент для моніторингу хмарних сервісів, вебсайтів та додатків. Вигляд інтерфейсу Site24x7 має вид [34]:

Рисунок 2.14 – Вид інтерфейсу Site24x7

Переваги [35]:

– підтримка плагінів;

– легкість використання.

Недоліки [35]:

– розуміння інтерфейсу та обмеженість функцій;

Sensu – інструмент моніторингу з відкритим кодом, призначений для моніторингу програм та служб з можливістю інтеграції з різними середовищами. Вигляд інтерфейсу Sensu має вид [36]:

Рисунок 2.15 – Вид інтерфейсу Sensu

Переваги [37]:

– спільнота;

– підтримка API.

Недоліки [37]:

– налаштування;

– конфігурація.

ManageEngine OpManager – програмне забезпечення для моніторингу мережі, серверів та структур з графічною панеллю керування. Вигляд інтерфейсу ManageEngine OpManager має вид [38]:

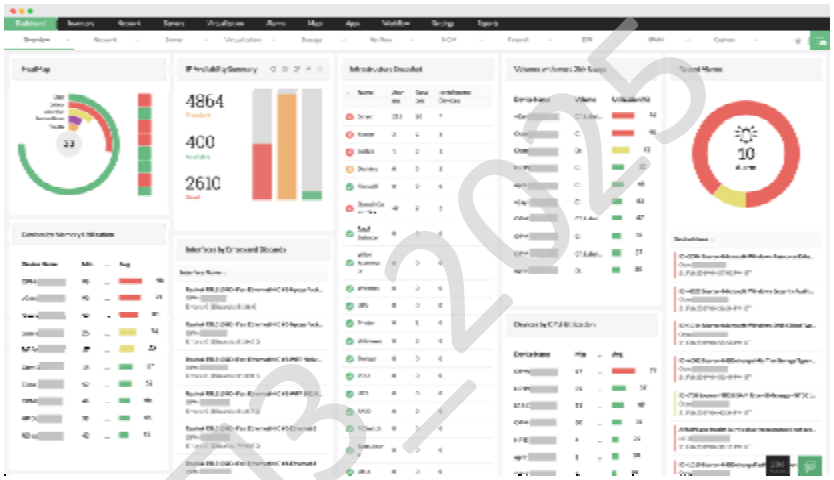

Рисунок 2.16 – Вид інтерфейсу ManageEngine OpManager

Переваги [39]:

– моніторинг у реальному часі;

– вартість;

– сповіщення.

Недоліки [39]:

– налаштування;

– обмеженість функцій.

WhatsUp Gold – інструмент моніторингу мережі з зручним сповіщенням. Вигляд інтерфейсу WhatsUp Gold має вид [40]:

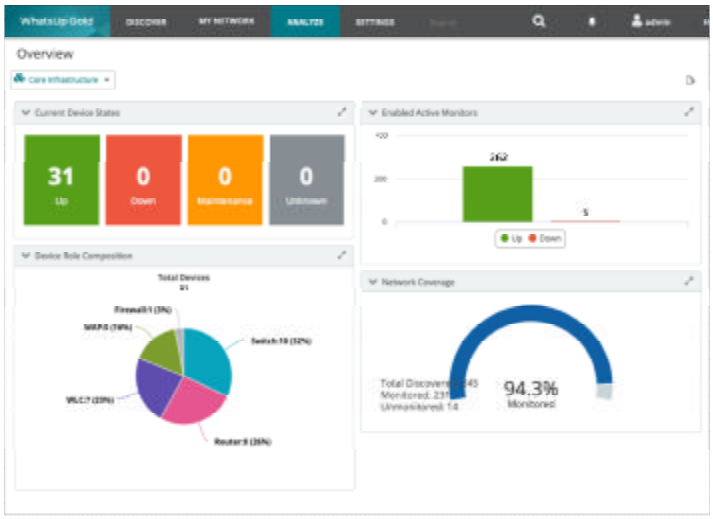

Рисунок 2.17 – Вид інтерфейсу WhatsUp Gold

Переваги [41]:

– простий інтерфейс;

– налаштування.

Недоліки [41]:

– висока вартість;

– інтеграція з інструментами.

New Relic – хмарна платформа для моніторингу за додатками, інфраструктурою, браузером та мобільними клієнтами. Вигляд інтерфейсу New Relic має вид [42]:

Рисунок 2.18 – Вид інтерфейсу New Relic

Переваги [43]:

– підтримка APM;

– спільнота;

– звітність.

Недоліки [43]:

– мобільний додаток;

– нестабільність роботи додатку.

BMC TrueSight Operations Management – платформа для контролювання програм з єдиного інтерфейсу. Вигляд інтерфейсу BMC TrueSight Operations Management має вид [44]:

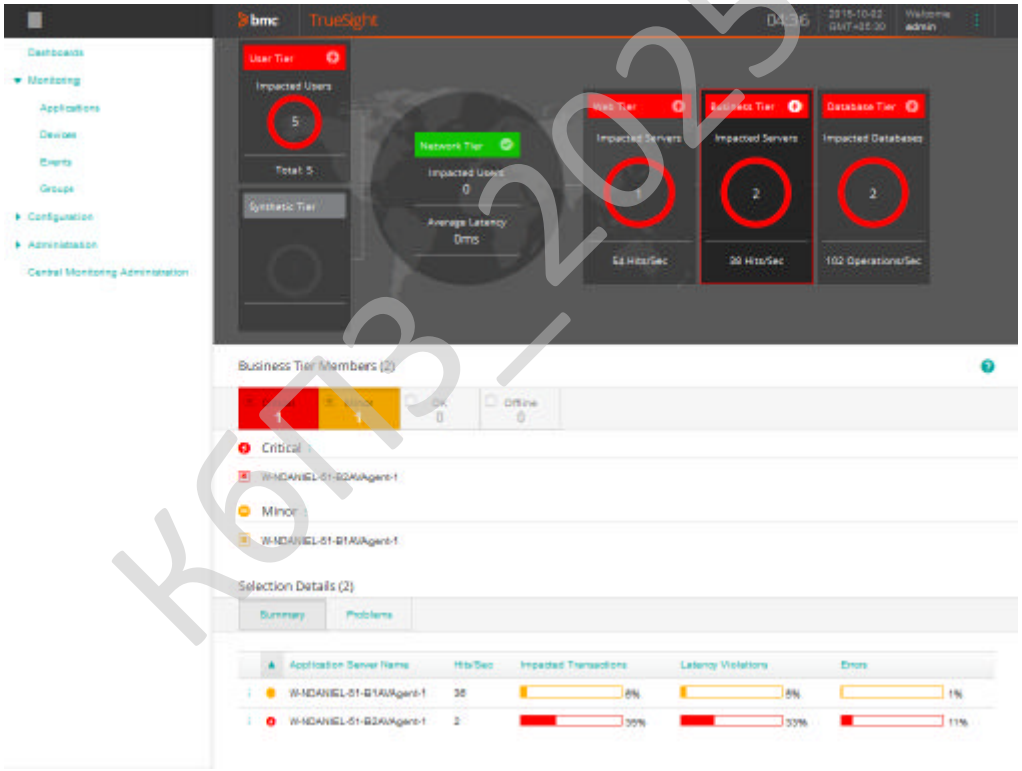

Рисунок 2.19 – Вид інтерфейсу BMC TrueSight Operations Management

Переваги [45]:

– налаштування;

– масштабованість.

Недоліки [45]:

– обмеженість функцій;

– складність використання.

Zenoss Core – платформа для моніторингу та управління мережею, створена на основі Zope. Zenoss Core надає інтерфейс що дозволяє контролювати доступність, продуктивність, конфігурацію та різні події в пристроях мережі. Вигляд інтерфейсу Zenoss Core має вид [46]:

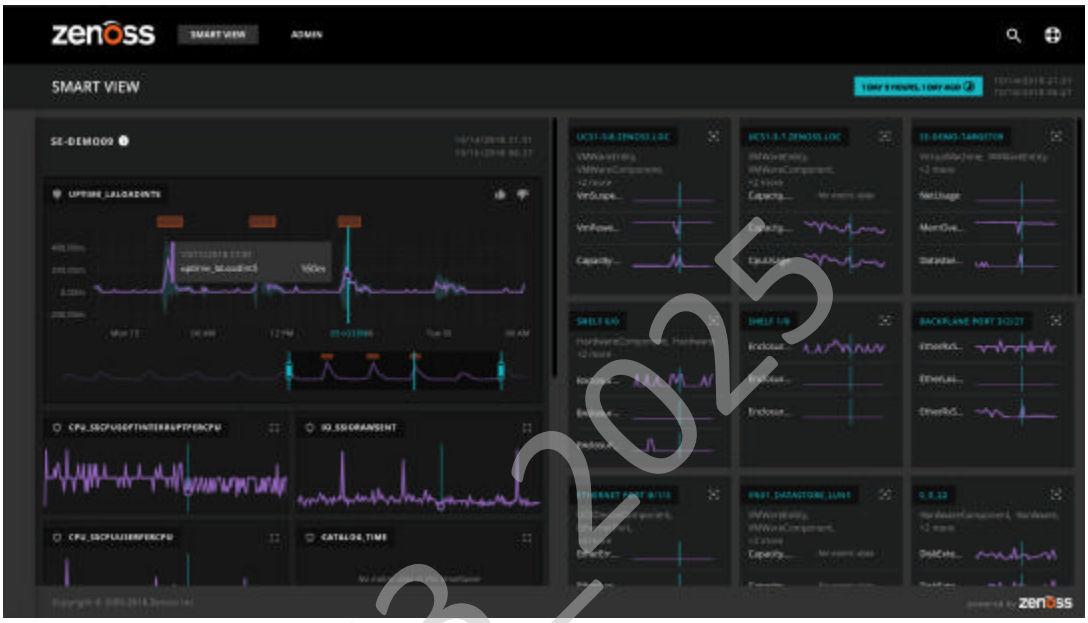

Рисунок 2.20 – Вид інтерфейсу Zenoss Core

Переваги [47]:

– гнучкий механізм виявлення;

– підтримка сучасних технологій;

– сповіщення.

Недоліки [47]:

– інтеграція з інструментами;

– налаштування;

– обмеженість функцій.

Centreon – система моніторингу, яка дозволяє контролювати інфраструктури та додатки. Вигляд інтерфейсу Centreon має вид [48]:

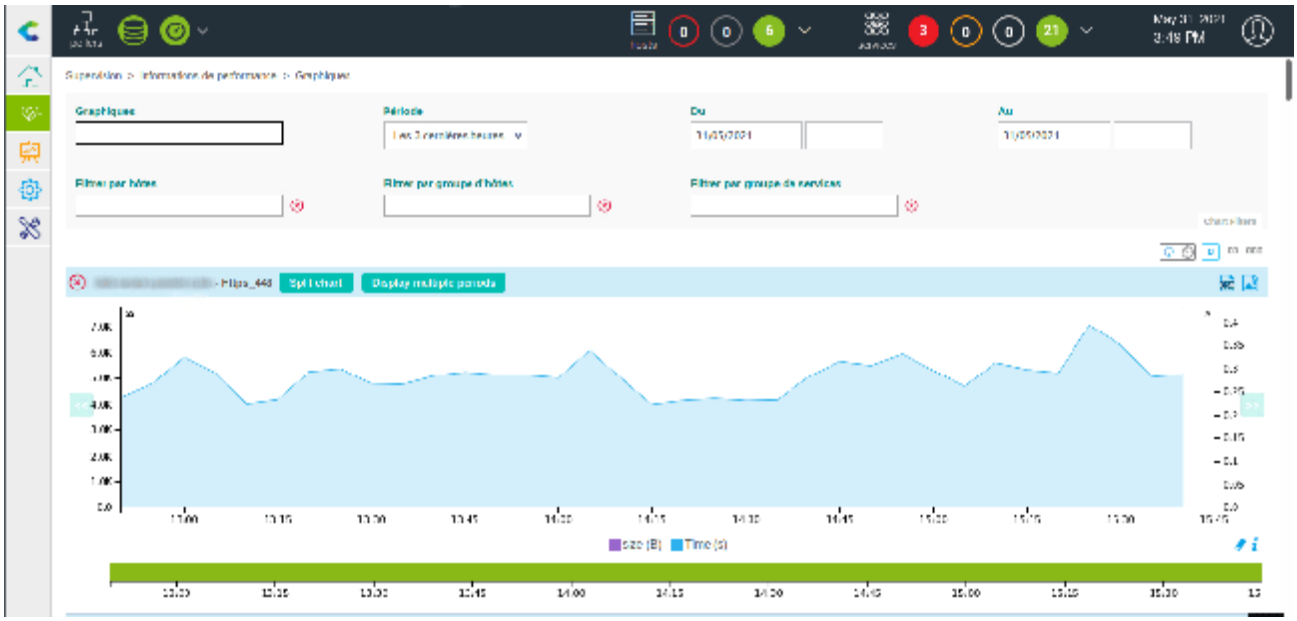

Рисунок 2.21 – Вид інтерфейсу Centreon

Переваги [49]:

– масштабованість;

– налаштування;

– наявність додаткових функцій;

– контроль необмеженої кількості пристроїв;

– гнучкість.

Недоліки [49]:

– безпека;

– відсутність API;

– моніторинг сервісів у хмарних середовищах.

LogicMonitor – платформа для моніторингу інфраструктури, додатків та мереж. Вона забезпечує видимість мереж, серверів, хмарних служб і програм у реального часу, дозволяючи керувати продуктивністю та доступністю системи. Завдяки архітектурі LogicMonitor спрощує розгортання та плавно масштабується разом із зростанням організації, підтримуючи великий спектр технологій за допомогою інтеграції та інформаційних панелей. Вигляд інтерфейсу LogicMonitor має вид [50]:

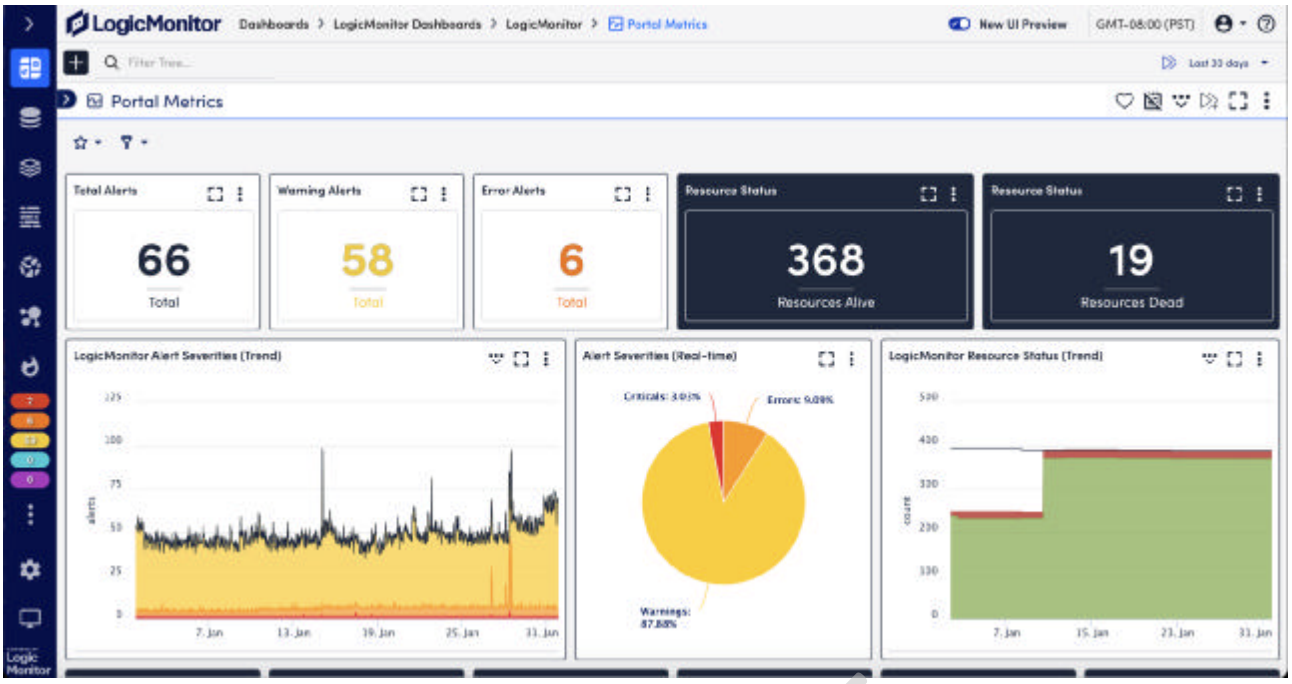

Рисунок 2.22 – Вид інтерфейсу LogicMonitor

Переваги [51]:

– звітність;

– сповіщення;

– візуалізація;

– ефективність;

– налаштування.

Недоліки [51]:

– моніторинг продуктивності;

– доступність;

– оновлення;

– обмеженість функцій;

– недостатність документації.

EventSentry – пакет моніторингу для Windows, який відстежує події, логи та стан системи. Додаток складається з таких основних компонентів: консоль керування, EventSentry, Heartbeat, мережеві служби, колектор, веб-звітність. Вигляд інтерфейсу EventSentry має вид [52]:

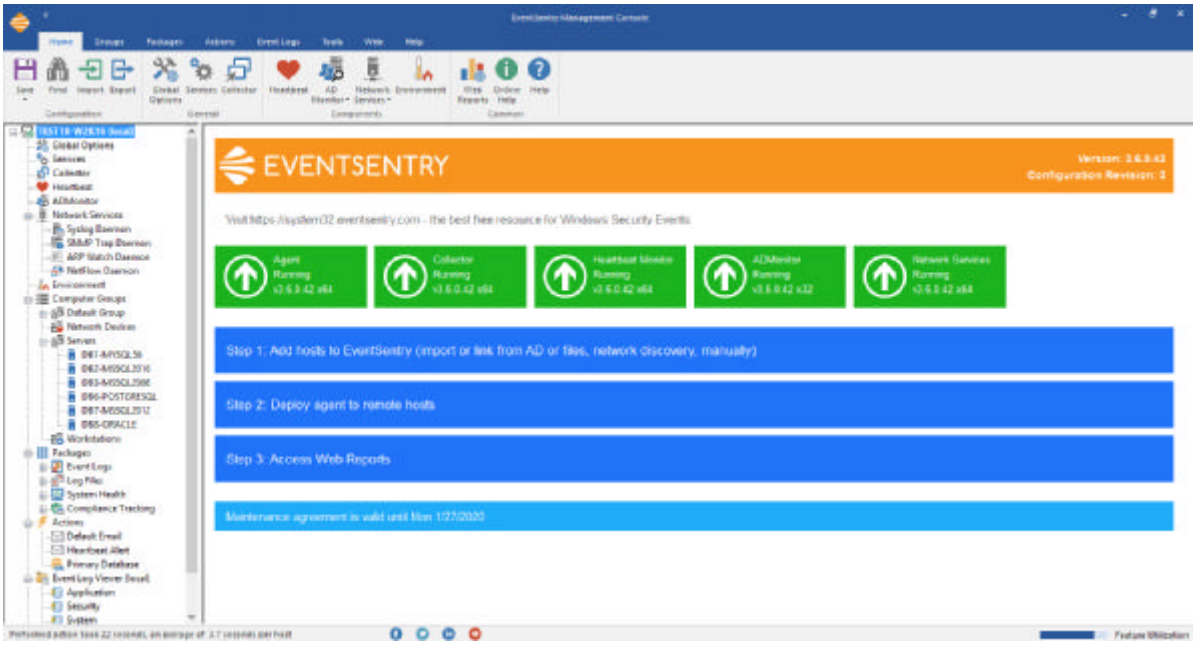

Рисунок 2.23 – Вид інтерфейсу EventSentry

Переваги [53]:

– легкість використання;

– зручність інтерфейсу;

– сповіщення;

– інформування;

– спільнота.

Недоліки [53]:

– налаштування;

– складність використання;

– проблеми з інструментами;

– відсутність API.

Pandora FMS – система моніторингу, яка призначена для відстеження продуктивності та проблем з роботою сервісів у мережах. Вона пропонує виявлення та визначення топології мережі. Створення модулів спрощено, а також підтримується досить великий набір протоколів. Архітектура є досить відкритою та масштабованою. Вигляд інтерфейсу Pandora FMS має вид [54]:

Рисунок 2.24 – Вид інтерфейсу Pandora FMS

Переваги [54]:

– графічні звіти;

– підтримка протоколів;

– відкритий код;

– виявлення та визначення топології мережі;

– гнучкість;

– підтримка протоклів.

Недоліки [54]:

– навчання системи;

– відсутність API;

– технічна підтримка.

Cabot – платформа для моніторингу інфраструктури з відкритим кодом, яка надає деякі з найкращих функцій PagerDuty, Server Density, Pingdom, Nagios та забезпечує надсилання SMS-повідомлень або сповіщень. Вигляд інтерфейсу Cabot має вид [55]:

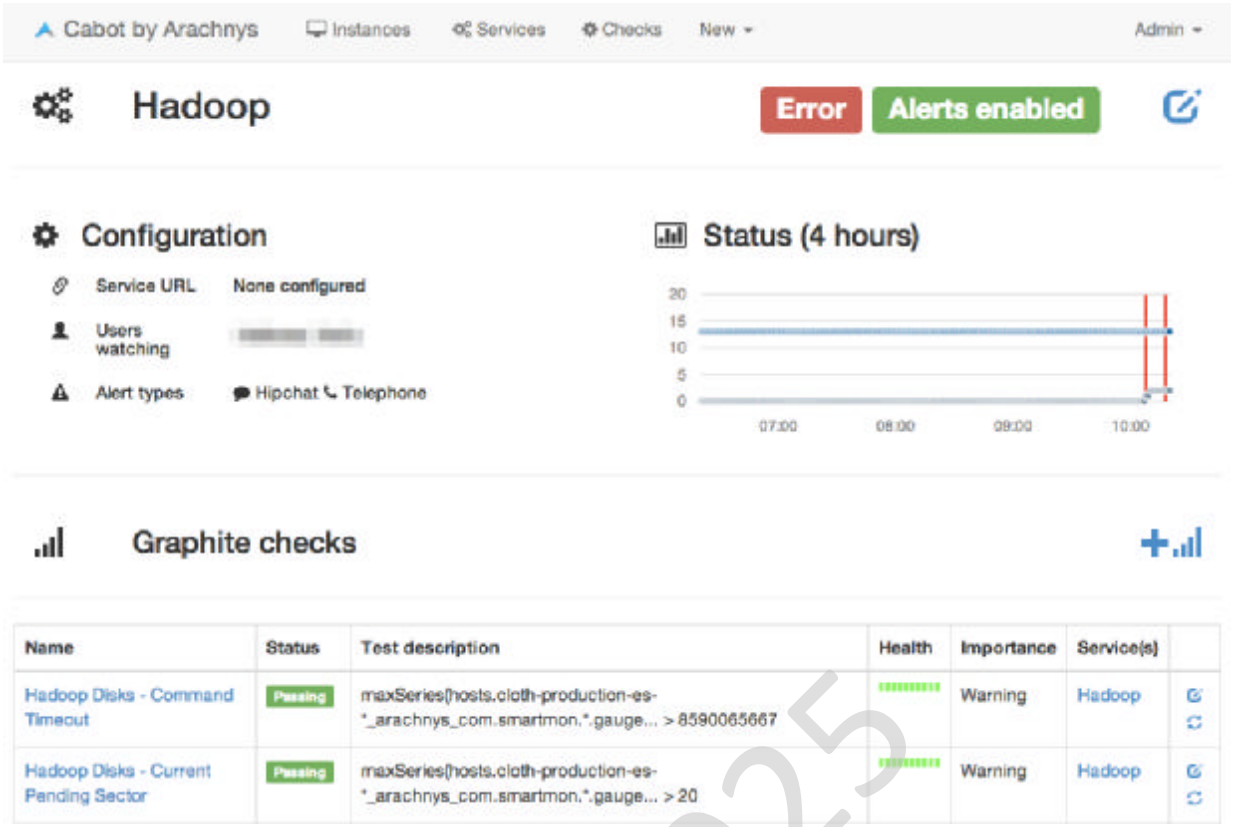

Рисунок 2.25 – Вид інтерфейсу Cabot

За допомогою Cabot можна запрограмувати пристрій системи EasyBUS та перевірити всі функції та команди даної системи. Також Cabot дозволяє користувачеві програмувати систему з великими перевагами відносно простоти, перевірок в автономному режимі та значного скорочення робочого часу. Cabot підтримує оповіщення та логіку сповіщень, що робить його більш зручним інструментом для моніторингу й контролю працездатності систем. Його гнучкість і відкритий код дозволяють інтегрувати додаткові можливості під потреби середовища [56].

Переваги [57]:

– легкість використання;

– налаштування;

– сповіщення.

Недоліки [57]:

– обмеженість функцій;

– недостатність документації;

– спільнота.

Opsview – платформа, яка використовується для моніторингу хмарних технологій та різної інфраструктури. Вона стабільна, добре працює та ефективно задовольняє відповідні вимоги моніторингу. Користувачі можуть клонувати різні сервіси для ефективного моніторингу з тестових налаштувань та зменшити кількість непотрібних сповіщень. Вигляд інтерфейсу Opsview має вид [58]:

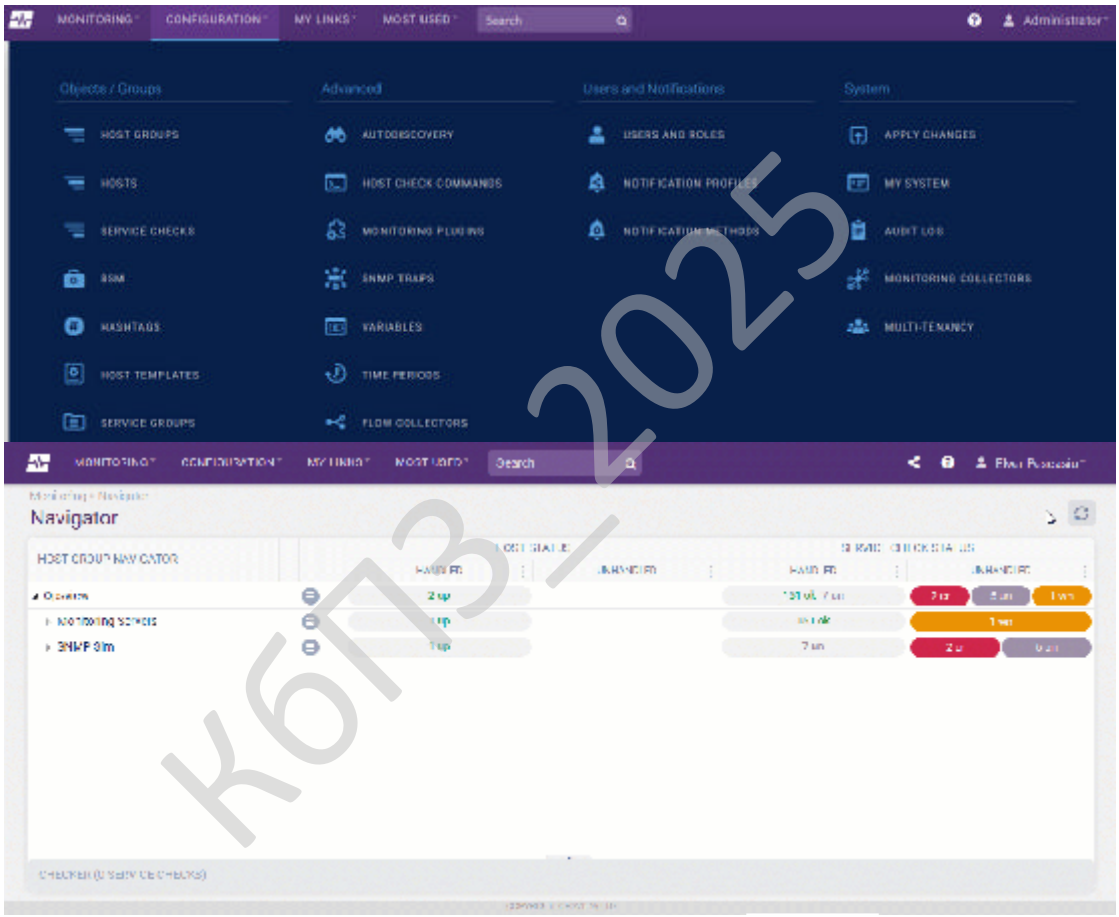

Рисунок 2.26 – Вид інтерфейсу Opsview

Переваги [58]:

– моніторинг хмарних платформ;

– клонування;

– фільтрування сповіщень.

Недоліки [58]:

– оновлення;

– звітність.

Існуючі системи пропонують великий спектр інструментів для моніторингу та безпеки. Проте жодна з розглянутих систем не надає одночасно таких опцій:

– гнучкої конфігурації;

– механізмів перерозподілу ресурсів;

– вбудованих засобів логування, оповіщення та безпеки;

– забезпечення моніторингу.

Таким чином, розробка системи дозволяє:

– адаптувати функціонал під потреби користувача;

– підвищити продуктивність;

– впровадити підхід до моніторингу та безпеки.

У рамках даної роботи було прийнято рішення про створення системи, що забезпечує відповідні функції в одному рішенні, з використанням мови програмування Python, яка дозволяє швидко реалізувати необхідне функціонування з мінімальною кількістю ресурсів.

Python – мова програмування високого рівня, яка поєднує простоту синтаксису з великими функціональними можливостями. Python дозволяє створювати модулі та компоненти, що сприяє підвищенню повторного використання коду. Python підтримується більшістю сучасних операційних систем, а стандартна бібліотека містить перелік інструментів для обробки файлів, роботи з мережею та управління процесами. Інтерпретатор доступний у відкритому вигляді, що дозволяє адаптувати його під різні задачі.

Python має потужну екосистему бібліотек та фреймворків, які спрощують розробку специфічних функцій. Дана мова програмування також відзначається широкими можливостями інтеграції з іншими мовами та технологіями. Завдяки своїм характеристикам, Python добре підходить для створення ефективних рішень, що забезпечують безпеку і стабільність системи.

Дизайн мови Python орієнтований на принципі об'єктно-орієнтованого програмування. Особливістю реалізації ООП в Python є поєднання простоти з функціоналом. Ця реалізація вирізняється зручністю використання в порівнянні з іншими об'єктно-орієнтованими мовами [59].

Особливості Python:

– множинне успадкування;
– поліморфізм;
– інкапсуляція;
– перевантаження операторів;
– імітація поля за допомогою функцій;
– управління доступу до полів;
– методи для управління найпоширенішими операціями;
– метапрограмування;
– повна інтроспекція;
– класові та статичні методи, класові поля.

Підтримка поліморфізму та множинного успадкування дає змогу реалізовувати більш складні архітектури без дублювання коду. Механізми інкапсуляції та управління доступом до полів дозволяють зберігати внутрішню логіку програми прихованою від користувачів. Ці особливості роблять Python надзвичайно потужним інструментом для розробки різноманітних програмних рішень та сприяють легшій підтримці та масштабуванню програм, що дозволяє швидко адаптувати код до нових вимог.

Однією з ключових характеристик Python є підтримка динамічної типізації, що дозволяє уникати обов'язкової вказівки типів даних під час написання коду. Мова також має велику кількість бібліотек для різних застосунків, що зменшує час на розробку та тестування. Більшість великих компаній та організацій використовують Python у своїй розробці, завдяки його масштабованості та високому рівню абстракції, що дозволяє зосередитись на вирішенні бізнес-проблем.

Основні переваги мови програмування:

– чистий синтаксис;

– відкритий код;

– переносність програм;

– стандартний дистрибутив з корисними модулями;

– зручність для розв'язання математичних проблем.

Основні недоліки мови програмування:

– продуктивність;

– низька швидкодія;

– глобальне блокування інтерпретатора;

– відсутність статичної типізації;

– обмежена підтримка мобільних платформ.

Таким чином, Python є ідеальним вибором для розробки системи кібербезпеки моніторингу серверів, що забезпечує своєчасний перерозподіл ресурсів та сприяє збереженню цілісності інформації [60].

Бібліотеки Python є важливою частиною цієї мови, забезпечуючи її потужними інструментами для реалізації різноманітних функцій і задач. Бібліотеки Python охоплюють широкий спектр застосувань, від обробки даних і аналізу статистики до машинного навчання.

Основні бібліотеки Python:

Math – забезпечує доступ до математичних функцій, визначених стандартом C.

Time – забезпечує різні функції, пов’язані з часом.

Os – забезпечує портативний спосіб використання залежних від операційної системи функцій.

Pathlib – пропонує класи, що представляють шляхи файлової системи із семантикою, відповідною до різних операційних систем.

Stat – визначає константи та функції для інтерпретації результатів.

Datetime – надає класи для роботи з датами та часом.

Struct – перетворює значення Python на структури C.

String – містить різноманітні функції та константи для роботи з рядками.

Json – забезпечує простий спосіб роботи з форматом JSON, що використовується для обміну даними між клієнтами та серверами.

Csv – дозволяє читати та записувати дані у форматі CSV.

Re – надає інструменти для роботи з регулярними виразами.

Numpy – забезпечує підтримку для великих, багатовимірних масивів і матриць, а також містить функції для виконання різноманітних математичних операцій над цими структурами.

Matplotlib – забезпечує візуалізацію даних.

Psutil – інструмент для моніторингу та управління системними ресурсами.

Smtplib – стандартна бібліотека для роботи з протоколом SMTP, що використовується для надсилання електронних листів.

Pandas – забезпечує структури даних та інструменти для роботи з даними, включаючи можливості для аналізу та маніпуляцій з табличними даними.

Requests – дозволяє легко здійснювати HTTP-запити та працювати з різними ресурсами.

Beautiful Soup – бібліотека для парсингу HTML і XML документів, що спрощує обробку та витяг даних з сторінок.

Flask – фреймворк для розробки, який дозволяє швидко створювати додатки на Python.

Django – фреймворк, що забезпечує швидкий розвиток додатків з готовими компонентами для безпеки, бази даних.

TensorFlow – бібліотека для машинного навчання, яка надає широкі можливості для побудови та навчання нейронних мереж.

Scikit-learn – бібліотека для машинного навчання, яка містить алгоритми для класифікації, регресії, кластеризації та обробки даних.

OpenCV – бібліотека для комп'ютерного зору, що містить функції для обробки зображень та відео.

SciPy – надає математичні та наукові функції, зокрема для оптимізації, інтеграції та інтерполяції.

Pillow – бібліотека для обробки зображень, що забезпечує можливості для відкриття, маніпуляцій і збереження зображень у різних форматах.

SQLAlchemy – бібліотека для роботи з базами даних, яка забезпечує ORM для зручної роботи з реляційними базами даних.

Pygame – бібліотека для створення відеоігор, що надає доступ до графіки, звуку та управління введенням.

Keras – API для нейронних мереж.

Python має безліч потужних бібліотек, які спрощують розробку різноманітних додатків. Ці бібліотеки дають можливість ефективно вирішувати складні завдання, дозволяючи їм зосереджуватися на основних аспектах логіки програми, що підвищує продуктивність та якість розробки [60, 61].

Основні можливості Python:

- синтаксична можливість для анотації параметрів;
- нова підсистема вводу-виводу;
- вирази для словників і множин;
- ієрархія типів для чисел;
- видалення деяких застарілих можливостей;
- реорганізація стандартної бібліотеки;
- підтримка асинхронного програмування;
- інтерактивний режим;
- розширені можливості для роботи зі списками;
- система типізації;
- контекстні менеджери;
- гнучка обробка винятків;
- вбудовані структури даних;
- інтеграція з іншими мовами;
- машинне навчання та штучний інтелект.

Python став одним із найпопулярніших виборів серед розробників у різних сферах. Він активно використовується в наукових дослідженнях, аналізі даних, веб-розробці, автоматизації задач, штучному інтелекті та машинному навчанні. Зростаюча спільнота користувачів і розробників постійно вдосконалює мову, що дозволяє їй залишатися на передовій технологій.

Завдяки своїй простоті, читабельності коду та широкому спектру бібліотек, Python залишається незамінним інструментом для користувачів. Таким чином, мова продовжує еволюціонувати, адаптуючись до сучасних викликів і потреб, що забезпечує її актуальність та значущість у галузі інформаційних технологій [60, 62].

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

При розробці програмного забезпечення системи моніторингу серверів з метою вчасного перерозподілу ресурсів було переглянуто кілька мов програмування, інструментів та середовищ, що можуть бути використані для реалізації модулів. Основними критеріями вибору були: гнучкість розробки, підтримка бібліотек для моніторингу, безпеки та веб-інтерфейсу, швидкість реалізації, доступність документації, кросплатформеність.

Python було обрано як основну мову програмування з таких причин:

1) Велика кількість бібліотек [61]:

– psutil - для збирання системних метрик;

– flask - для побудови веб-інтерфейсу;

– smtplib - для відправки email-сповіщень;

– jwt, cryptography - для реалізації авторизації та захисту даних.

2) Швидкість розробки та зручний синтаксис: Python дозволяє створювати функціональні моделі за короткий час.

3) Кросплатформеність: код працює під Windows, Linux, macOS.

4) Документація: забезпечує швидке розв’язання проблем, які виникають у процесі розробки та надає довідкову інформацію.

5) Інтеграція з іншими система.

Вибір архітектури та технологій [61]:

1) Flask – дозволяє швидко реалізувати інтерфейс, авторизацію та візуалізацію логів.

2) JSON – простий формат для зчитування, взаємодії між модулями та конфігурації.

3) Threading + signal – використовується для запуску модулів моніторингу, з можливістю обробки сигналів завершення.

4) SMTP – використовується для реалізації сповіщень.

5) JWT – використовується як спосіб безпечної авторизації користувачів при взаємодії з інтерфейсом та забезпечує захист даних.

Середовище розробки:

1) Visual Studio Code – середовище з підтримкою різних розширень для Python, автодоповнення коду та інтеграції з Git.

2) Python 3.10 – версія з підтримкою сучасних конструкцій.

Таким чином, обрані засоби і мова програмування дозволяють побудувати надійну, гнучку та розширювану систему, що відповідає вимогам безпеки, продуктивності та зручності для користувача.

2.3 Розгорнута постановка завдання

Метою роботи є розробка програмного забезпечення системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації, яке дозволить здійснювати збір, обробку, аналіз та реагування на критичні ситуації, пов’язані з навантаженням на серверні ресурси, з можливістю інформування користувачів та забезпечення інформаційної безпеки системи.

Для досягнення поставленої мети необхідно реалізувати такі основні функціональні компоненти:

1) Модуль моніторингу серверів – забезпечує періодичне зчитування метрик системи: завантаження CPU, використання оперативної пам’яті, дискового простору, мережевої активності.

2) Модуль аналізу даних – відповідає за обробку зібраної інформації, порівняння її з пороговими значеннями та виявлення ознак перевантаження або аномальної поведінки.

3) Модуль розподілу ресурсів – автоматично виконує заходи з оптимізації навантаження.

4) Модуль сповіщення – реалізує надсилання повідомлень електронною поштою у разі виявлення критичних подій.

5) Модуль логування – веде журнал усіх подій системи, зберігаючи інформацію про всі операції, збої, перевантаження та втручання.

6) Модуль управління конфігураціями – дозволяє змінювати параметри роботи системи через інтерфейс.

7) Модуль безпеки – включає механізми виявлення DDoS-атак, захисту переданої інформації, а також механізми аутентифікації та авторизації користувачів через JWT токени.

8) Модуль тестування – передбачає написання юніт-тестів для перевірки коректності роботи всіх функціональних блоків.

9) Модуль документації – включає алгоритм дій користувача для зручного користування.

Таким чином, реалізація зазначених модулів забезпечить виконання завдань із моніторингу, безпеки та керування серверними ресурсами, а також відповідатиме вимогам випускної кваліфікаційної роботи.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розроблена система моніторингу серверів з функціями кібербезпеки реалізує комплексний підхід до забезпечення надійності, стабільності та захисту серверної інфраструктури. Її функціонування базується на послідовній взаємодії модулів, які виконують визначені ролі та взаємодіють між собою у реальному часі.

Основні етапи функціонування системи:

1) Ініціалізація та конфігурація:

Після запуску система завантажує початкові налаштування з конфігураційного файлу, який містить такі параметри:

– порогові значення;

– інтервал моніторингу.

2) Моніторинг серверів:

Модуль моніторингу періодично зчитує метрики системи:

– рівень використання центрального процесора;

– обсяг оперативної пам’яті;

– заповнення дискового простору;

– обсяг вхідного мережевого трафіку.

– обсяг вихідного мережевого трафіку

Отримані метрики зберігаються у лог-файлі.

3) Аналіз даних:

Модуль аналізу порівнює отримані метрики з заданими пороговими значеннями. У разі перевищення системних лімітів виявляється стан перевантаження. Додатково реалізовано виявлення аномалій, які можуть вказувати на потенційні DDoS-атаки.

Результат аналізу містить:

– статус;

– список перевищених параметрів;

– прапор критичності.

4) Розподіл ресурсів:

У разі навантаження активується модуль розподілу ресурсів, який виконує наступні дії:

– очищення тимчасових файлів;

– завершення процесів;

– оптимізація використання пам’яті, дискового простору та CPU.

5) Система сповіщень:

Після виявлення небезпечного стану або атаки, модуль сповіщення надсилає повідомлення на електронну пошту з описом проблеми.

6) Логування:

Модуль логування веде запис усіх операцій, повідомлень, подій моніторингу, дій користувача та сповіщень. Через веб-інтерфейс користувач має змогу: перегляди, очищати історію логів та перевіряти статус відповідних серверів.

7) Безпека та доступ:

Доступ до системи обмежено механізмами авторизації з використанням JWT-токенів. Дані шифруються, паролі захищені. Модуль додатково фільтрує підозрілу активність.

8) Тестування та документація:

Модулі tt.py та dc.py забезпечують підтримку проєкту. За допомогою юніт-тестів перевіряється коректність ключових функцій, а користувач може переглядати покрокову інструкцію.

Система працює як єдиний взаємопов’язаний комплекс. Вона здатна виявляти критичні навантаження, ефективно реагувати, забезпечувати безпеку та надавати зручний інтерфейс для керування та контролю.

3.2 Розробка структурної схеми

Структурна схема розробленої системи моніторингу серверів з метою вчасного перерозподілу ресурсів відображає основні компоненти системи та їх взаємозв’язки. Вона дозволяє зрозуміти логіку побудови самої програми та взаємодію його основних складових [63, 64].

Система побудована за модульним принципом, де кожен компонент відповідає за окрему функцію, а зв’язки між ними реалізуються через передачу даних.

Основні компоненти структурної схеми:

1) Користувач:

– взаємодіє із системою;

– виконує зміну налаштувань, перевірку логів, проходить авторизацію.

2) Модуль конфігурації:

– зберігає поточні налаштування;

– дозволяє редагувати параметри.

3) Модуль моніторингу:

– збирає метрики з ОС;

– передає дані до модуля аналізу;

– записує інформацію в лог-файл.

4) Модуль аналізу даних:

– оцінює отримані метрики;

– виявляє перевантаження або аномалії.

5) Модуль розподілу ресурсів:

– виконує очищення та оптимізацію при навантаженнях;

– описує дії у логах.

6) Модуль сповіщення:

– надсилає повідомлення при перевантаженнях.

7) Модуль логування:

– веде журнал подій;

– забезпечує перегляд та очищення логів.

8) Модуль безпеки:

– контролює доступ;

– виявляє підозрілу активність;

– забезпечує цілісність та захист даних.

9) Модулі тестування та документації:

– забезпечують стабільність і супровід системи.

Вигляд структурної схеми:

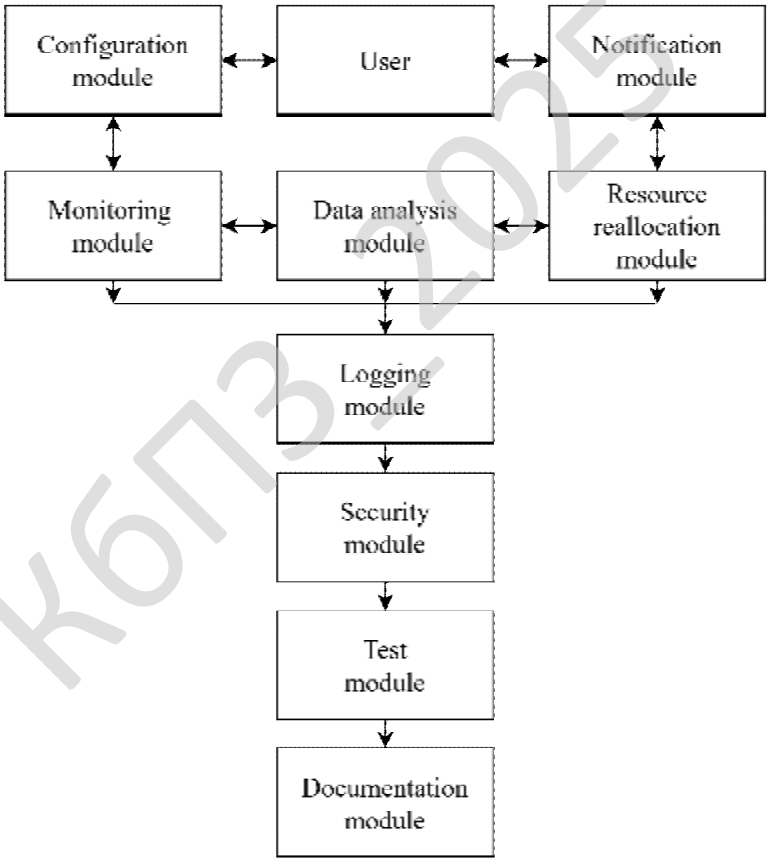

Рисунок 3.1 – Структурна схема

Структурна схема демонструє логічно завершену систему, в якій кожен модуль має чітке призначення, а взаємозв'язки забезпечують повноцінний цикл моніторингу, реагування, безпеки та зручного керування.

3.3 Розробка функціональної схеми

Функціональна схема системи моніторингу серверів з метою вчасного перерозподілу ресурсів відображає взаємодію всіх ключових модулів, що забезпечують роботу системи, її захист та зручність для користувача [65].

Основні функціональні модулі системи:

1) Модуль конфігурації:

Надає можливість налаштовувати параметри системи: інтервали перевірок, пороги перевантаження. Працює із конфігураційним файлом.

2) Модуль моніторингу:

Відповідає за періодичне збирання метрик стану серверів, таких як CPU, RAM, диск, мережа. Дані надходять до модуля аналізу даних для обробки.

3) Модуль аналізу даних:

Проводить аналіз отриманих метрик. У разі виявлення перевантаження викликає модуль перерозподілу ресурсів.

4) Модуль перерозподілу ресурсів:

Приймає рішення про перерозподіл навантаження. Передає результати на логування та сповіщення.

5) Модуль сповіщення:

Інформує осіб про події, що перевищують пороги. Повідомлення надсилаються за допомогою електронної пошти.

6) Модуль логування:

Фіксує усі дії системи. Дані зберігаються у зручному форматі.

7) Модуль безпеки:

Відповідає за перевірку, авторизацію користувачів, захист даних.

8) Модуль тестування:

Включає набір тестів для перевірки правильності роботи кожного модуля.

9) Модуль документації:

Надає покрокову інструкцію для роботи з програмою.

Вигляд функціональної схеми:

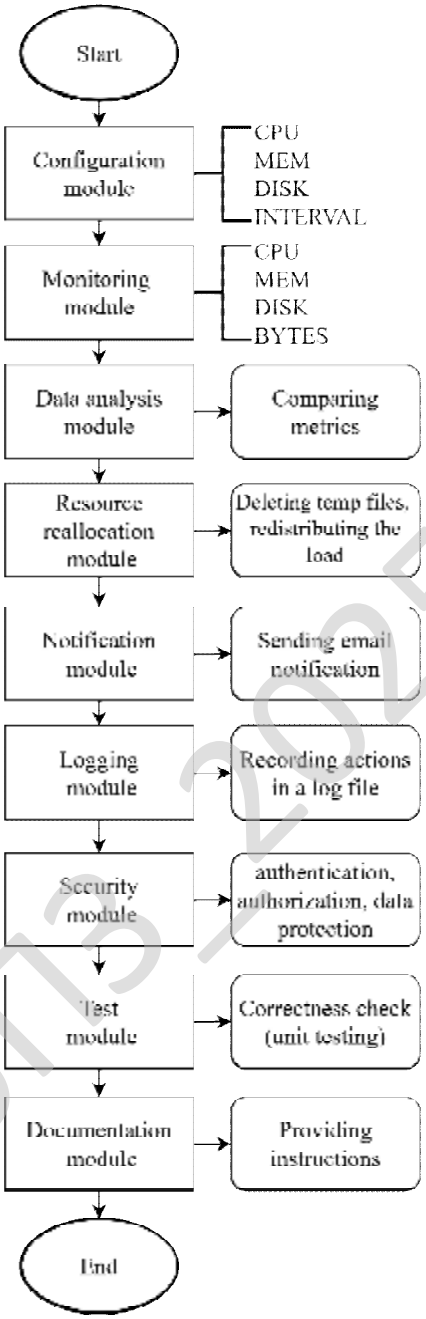

Рисунок 3.2 – Функціональна схема

Функціональна схема є важливою для розуміння логіки системи, дозволяючи відтворювати взаємодію між модулями та забезпечити функціонування всієї програми. Кожен з модулів відповідає за свою функцію.

3.4 Розробка діаграми процесів

Діаграма процесів демонструє послідовність дій у рамках роботи системи моніторингу серверів та реакції на перевантаження. Це дозволяє відслідковувати заданий алгоритм прийняття рішень, дій користувача, системи та кожного окремого модуля.

Крім того, діаграма процесів забезпечує уявлення про взаємодію між модулями, що дозволяє ефективніше виявляти логічні помилки або місця, які потребують більшої оптимізації. Завдяки цьому можна не лише краще зрозуміти архітектуру системи, а й спростити подальше тестування, налагодження та масштабування програмного забезпечення. Також діаграма процесів відіграє важливу роль у документуванні системи, що є корисним під час розробки, технічної підтримки та майбутньої модернізації [66].

Основні процеси в системі:

1) Запуск системи – ініціалізується головний цикл, забезпечується конфігурація.

2) Моніторинг метрик – зчитування та збір даних з інтервалом.

3) Аналіз даних – визначається, чи перевищено значення допустимих порогів.

4) Рішення:

– якщо навантаження в межах норми - запис у лог-файл;

– якщо перевищено - активуються модулі перерозподілу та сповіщення.

5) Логування дій – усі дії модулів фіксуються у відповідних текстових файлах.

6) Забезпечення безпеки - авторизація користувача, цілісність та захист даних.

7) Тестування та документація – перевірка правильності роботи модулів та надання інструкції щодо функціонування програми.

Вигляд діаграми процесів:

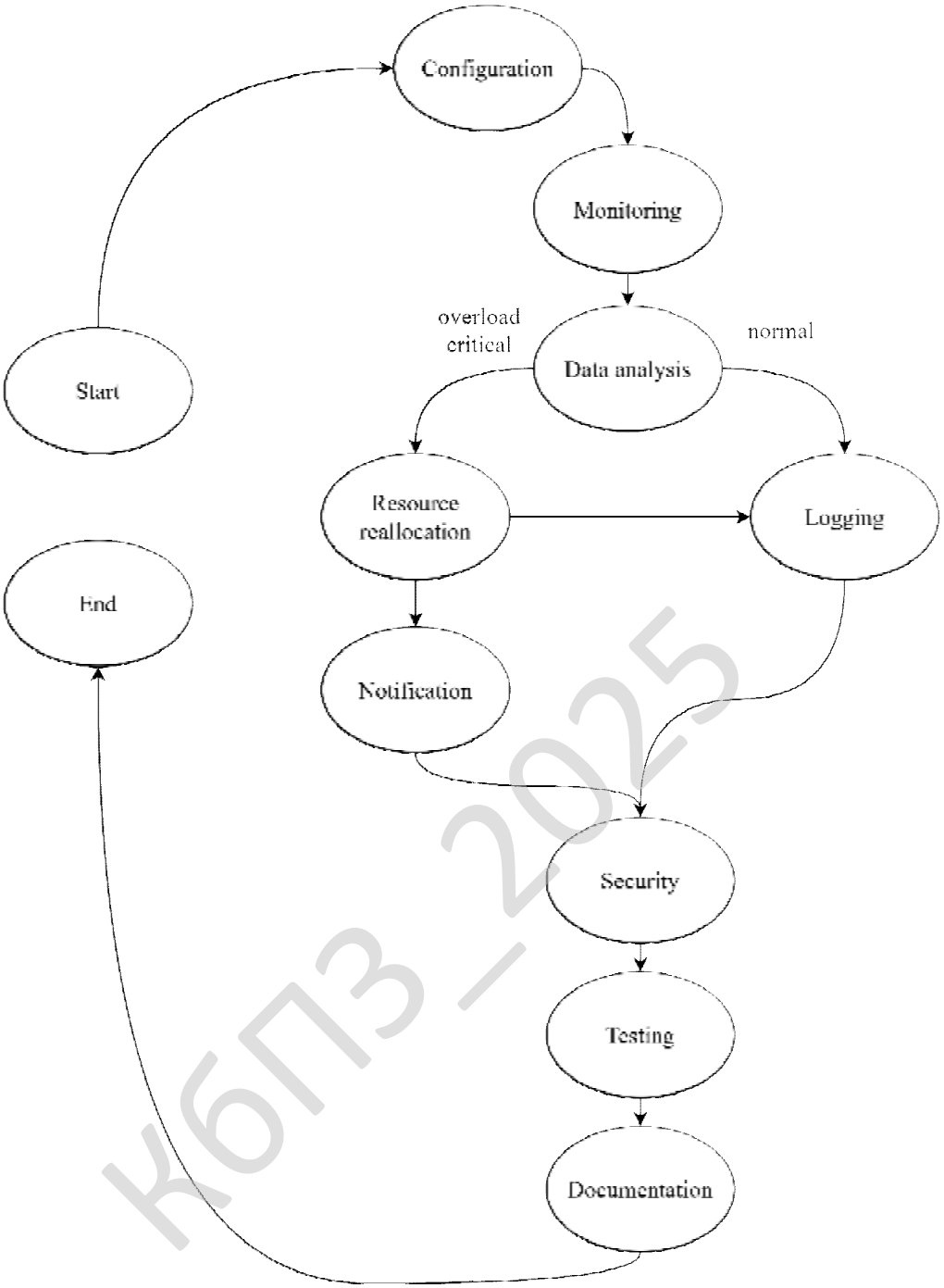

Рисунок 3.3 – Діаграма процесів

Діаграма процесів відображає ключові моменти роботи системи, логіку прийняття рішень та послідовність взаємодії між модулями. Це дає змогу структурувати роботу системи та зменшити кількість помилок під час її виконання.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Система моніторингу та перерозподілу ресурсів серверів реалізується на мові програмування Python. Кожен модуль виконує окрему функцію. Такий підхід дозволяє підтримувати та тестувати систему незалежно від інших компонентів.

Блок-схема основної програми:

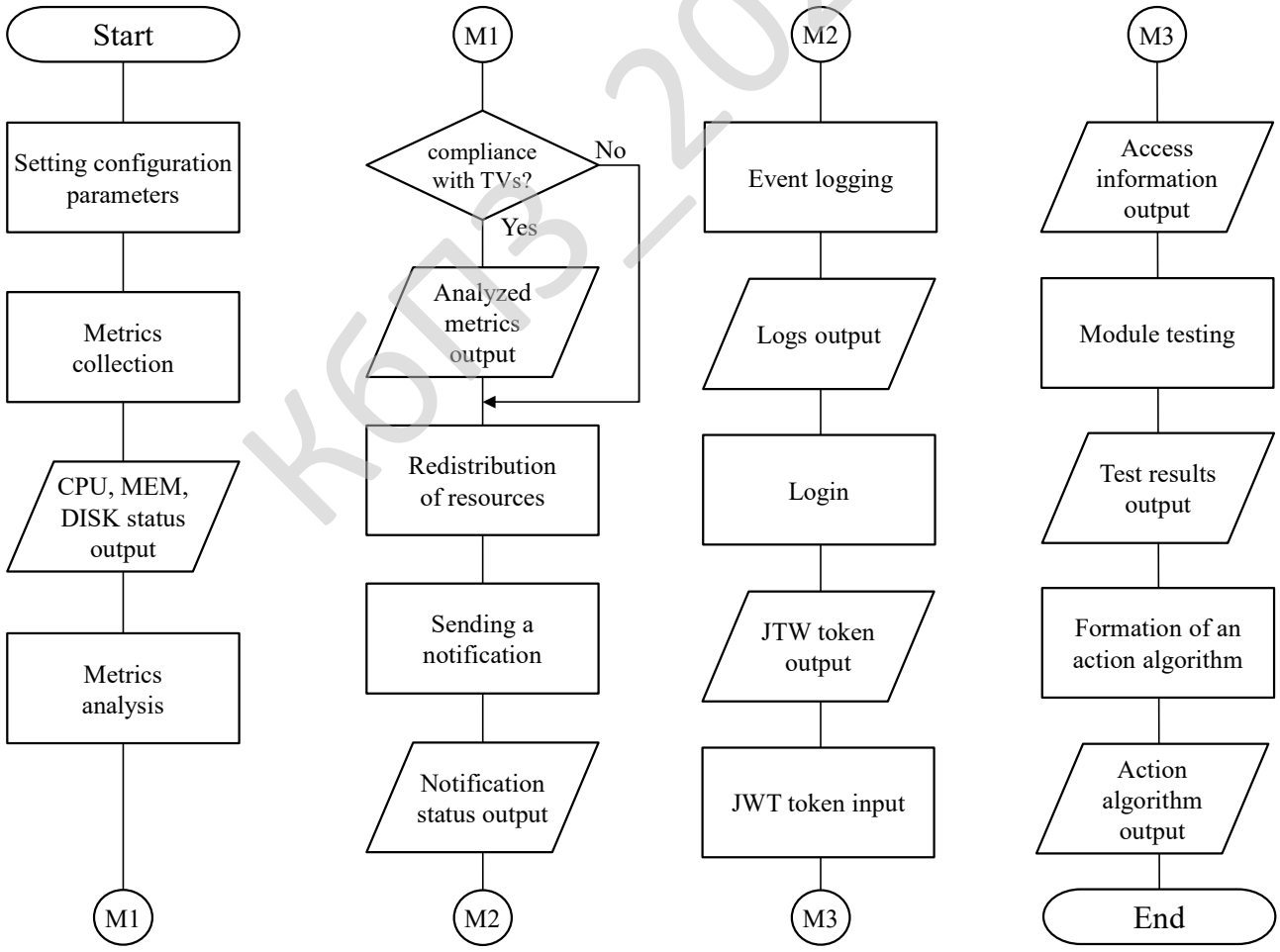

Рисунок 4.1 – Блок-схема основної програми

Блок-схема роботи підпрограми:

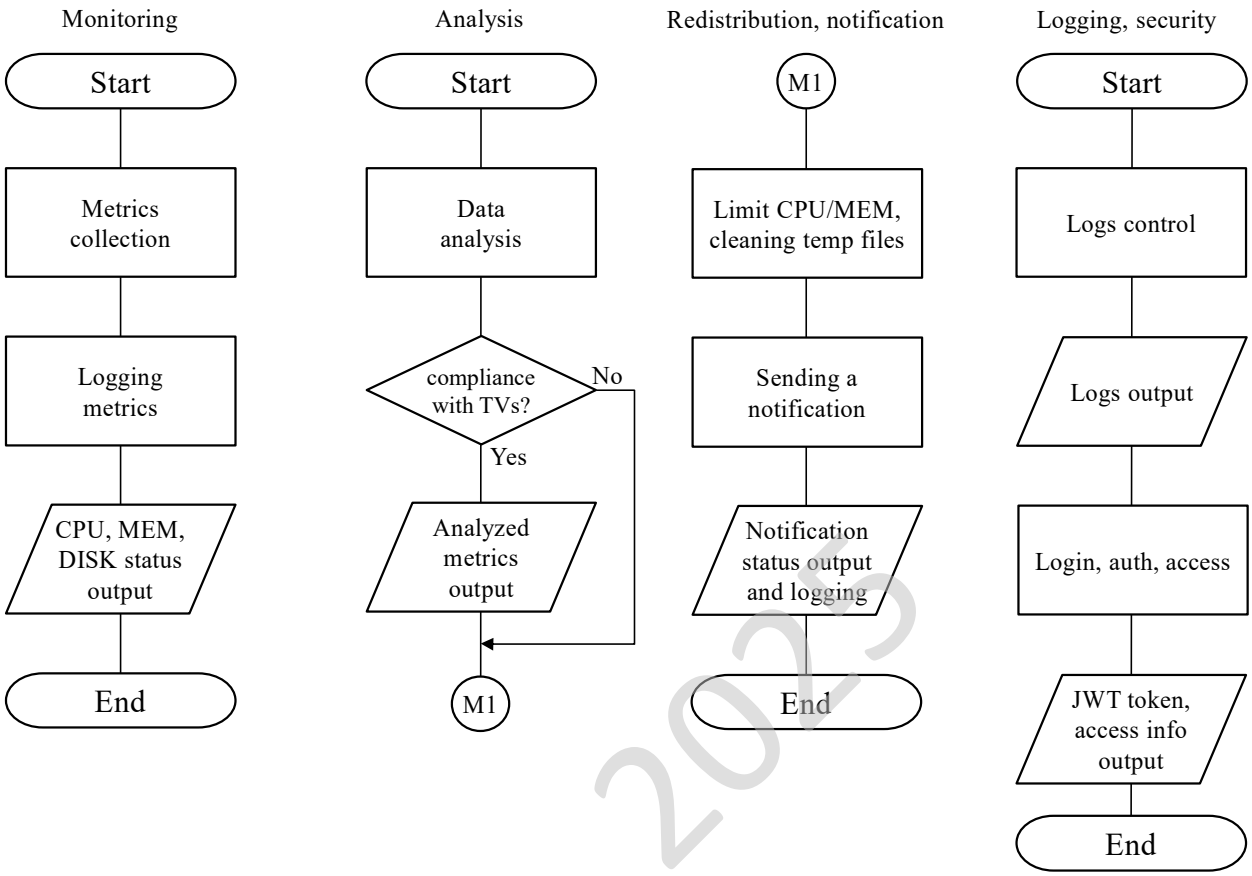

Рисунок 4.2 – Блок-схема роботи підпрограми

Алгоритм конфігурації:

1) Зчитування файлу конфігурації.

2) Застосування та зміна параметрів.

Алгоритм моніторингу:

1) Періодичний збір даних з використанням psutil.

2) Збереження метрик та запис інформації в лог-файл.

Алгоритм аналізу даних:

1) Аналіз метрик.

2) Виявлення перевантажень.

Алгоритм перерозподілу навантаження:

1) Перерозподіл навантажень.

2) Передавання даних до модулів сповіщення та логування.

Алгоритм оповіщення:

1) Отримання події.

2) Формування сповіщення.

3) Надсилання сповіщення на електронну пошту.

Алгоритм логування:

1) Отримання подій.

2) Формування записів.

3) Збереження в лог-файли.

Алгоритм безпеки:

1) Забезпечення цілісності та захисту даних.

2) Контроль доступу та виявлення підозрілої активності.

Алгоритм тестування:

1) Проведення юніт-тестів.

Алгоритм документації:

1) Відображення алгоритму дій.

Блок-схеми та алгоритми описують процес функціонування всієї системи моніторингу та перерозподілу ресурсів. Кожен модуль реалізує конкретну задачу та взаємодіє з іншими, що забезпечує стабільність, безпеку та гнучкість. Це дозволяє полегшити обслуговування системи, впровадити зміни та перевірити правильність реалізації кожного модуля. Система є готовою до використання та подальшого вдосконалення для різних сфер використання [67].

4.2 Опис використаних бібліотечних функцій та шаблонів

Задача отримання логів від серверів вимагає використання єдиного протоколу, який реалізується за допомогою модуля lg.py. Для встановлення зв’язку використано функцію disp_logs(). Формат виклику такої функції наведено в наступному фрагменті:

```
def disp_logs():
```

```
    try:
        with open('system_logs.log', 'r') as f:
            logs = f.readlines()
    return render_template_string(html_temp, logs=''.join(logs))
```

Наведена функція не вимагає приймання параметрів. Вона відкриває файл логів у режимі читання, зчитує всі рядки та передає їх для відображення. Її основна мета – завантажити вміст лог-файлу та повернути веб-сторінку.

Наведена функція повертає веб-сторінку у форматі HTML як рядок.

Задача отримання метрик від серверів вимагає використання єдиного протоколу, який реалізується за допомогою модуля mon.py. Для встановлення зв'язку використано функцію col_metr(). Формат виклику такої функції наведено в наступному фрагменті:

```
def col_metr(s):
    metrics = {
        'datetime': time.strftime('%Y-%m-%d %H:%M:%S'),
        'cpu_usage': psutil.cpu_percent(interval=1),
        'memory_usage': psutil.virtual_memory().percent,
        'disk_usage': psutil.disk_usage('/').percent,
        'network': {
            'bytes_sent': psutil.net_io_counters().bytes_sent,
            'bytes_recv': psutil.net_io_counters().bytes_recv
        }
    }
    return metrics
```

Наведена функція приймає системний стан, оскільки вона є методом класу Serversmonitoring. Вона не вимагає параметрів при виклику та виконується у циклі для збору метрик. Її основна мета – зібрати метрики стану системи.

Наведена функція повертає словник metrics, який містить зібрані дані про стан системи.

Задача аналізу даних від серверів вимагає використання єдиного протоколу, який реалізується за допомогою модуля ad.py. Для встановлення зв'язку використано функцію anldt(). Формат виклику такої функції наведено в наступному фрагменті:

```
metrics = mon.col_metr()
analysis_result = anser.anldt(metrics)
```

Наведена функція приймає метрики, які зібрані модулем mon.py для аналізу. Її основна мета – проаналізувати на перевантаження зібрані метрики.

Наведена функція повертає словник iss з оцінкою ситуації перевантаження та описом стану перевищення порогів.

Задача перерозподілу ресурсів від серверів вимагає використання єдиного протоколу, який реалізується за допомогою модуля ar.py. Для встановлення зв'язку використано функцію re_res(). Формат виклику такої функції наведено в наступному фрагменті:

```
metrics = mon.col_metr()
analysis_result = anser.anldt(metrics)
if analysis_result['overload']:
    actions = redor.re_res(metrics)
```

Наведена функція приймає метрики, які зібрані модулем mon.py і проаналізовані модулем ad.py. Її основна мета – забезпечити перерозподіл ресурсів, виконавши ряд відповідних дій: cpu_us(), mem_us(), disk_us().

Наведена функція повертає список дій, які були виконані для перерозподілу ресурсів.

Задача надсилання сповіщення від серверів вимагає використання єдиного протоколу, який реалізується за допомогою модуля no.py. Для встановлення зв'язку використано функцію send_emnot(). Формат виклику такої функції наведено в наступному фрагменті:

```
inc_metrics = "\n".join(analysis_result['desc'])
subject = G_NOTIFICATION[not_st]['subject']
rec = G_NOTIFICATION[not_st]['rec']
send_emnot(subject, inc_metrics, rec)
```

Наведена функція приймає тему листа, метрики та список отримувачів. Її основна мета – надіслати сповіщення на електронну пошту.

Наведена функція повертає вивід у консоль повідомлення про стан відправлення та надсилає сповіщення через сервіс Gmail.

Задача процесу конфігурації від серверів вимагає використання єдиного протоколу, який реалізується за допомогою модуля cn.py. Для встановлення зв'язку використано функцію disp_con(). Формат виклику такої функції наведено в наступному фрагменті:

```
def disp_con():
    config = con.g_conf()
    return render_template_string(html_temp, config=config)
```

Наведена функція не вимагає приймання параметрів. Вона викликає метод g_conf(), який повертає конфігурацію системи. Її основна мета – отримати конфігураційні параметри системи та повернути веб-сторінку для перегляду.

Наведена функція повертає веб-сторінку у форматі HTML як рядок.

Задача забезпечення безпеки від серверів вимагає використання єдиного протоколу, який реалізується за допомогою модуля cr.py. Для встановлення зв'язку використано функції login() та protected(). Формат виклику таких функцій наведено в наступному фрагменті:

```
def login():
    if request.method == 'GET':
        return '''
        <form action="/login" method="post">
        <fieldset style="width: fit-content; padding: 20px; border: 2px
 solid #ccc; border-radius: 10px;">
            <legend style="font-weight: bold;">Login</legend>
            <label>Username:</label><br>
            <input type="text" name="username"><br>
            <label>Password:</label><br>
            <input type="password" name="password"><br>
            <input type="submit" value="Login" style="margin-top: 10px;">
        </fieldset>
        </form>
        '''
    elif request.method == 'POST':
        username = data.get('username')
        password = data.get('password')
        if username in USER_DB and USER_DB[username] ==
        hashlib.sha256(password.encode()).hexdigest():
            return f'''
            <textarea readonly style="width: 100%; height:
            100px;">{token}</textarea>
            '''
        else:
            logging.warning("Failed login attempt for {}".format(username))
            return '<h1>Incorrect username or password</h1>', 401
def protected():
    if request.method == 'GET':
        return '''
        <form action="/protected" method="post">
            <input type="text" name="token" style="width: 100%;"><br><br>
            <input type="submit" value="Check access">
        </form>
        '''
    elif request.method == 'POST':
            return f'<h1>{user}</h1>', 401
        return f'<h1>Access allowed for {user}</h1>'
```

Наведені функції не вимагають приймання параметрів. Їх основна мета – забезпечити логін та надання доступу.

Наведені функції повертають JWT токен та стан доступу.

4.3 Захист розробленого програмного забезпечення

Захист програмного забезпечення є одним із ключових етапів у процесі розробки системи моніторингу серверів з метою вчасного перерозподілу ресурсів. Зважаючи на важливість оброблюваних даних, а також можливість несанкціонованого доступу до інтерфейсу, у розробленій системі реалізовано відповідні механізми захисту:

1) Аутентифікація та авторизація користувачів:

Реалізовано механізм автентифікації за допомогою логіна та пароля. Паролі передаються у зашифрованому вигляді, а після входу користувач отримує токен авторизації на основі якого відбувається надання доступу.

У системі застосовано JWT для створення токена доступу.

2) Шифрування та передача даних:

Для забезпечення конфіденційності обміну повідомленнями, а також для захисту метрик реалізовано алгоритм шифрування AES. Ключ генерується локально.

3) Виявлення підозрілої активності:

Система має вбудований механізм виявлення підозрілих подій: повторних спроб входу з неправильними даними та DDoS атак. У випадку перевищення певного порогу система надсилає сповіщення на електронну пошту.

4) Захист даних:

Дії, які пов’язані з моніторингом, фіксуються в логах.

5) Розмежування прав доступу:

У системі можлива реалізація декількох рівнів доступу. Це дозволяє обмежити небезпечні дії.

Механізми захисту забезпечують відповідний рівень безпеки для системи, дозволяючи запобігати несанкціонованому доступу, зберігати конфіденційність даних, а також реагувати на підозрілу активність. Це дозволяє перенести систему у середовище з підвищеними вимогами до інформаційної безпеки [68].

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблена система моніторингу серверів з метою вчасного перерозподілу ресурсів придатна до впровадження в середовище функціонування підприємства. Враховуючи модульну структуру, відкритість інтерфейсу та гнучкість у налаштуваннях, система може бути адаптована до різних сфер використання.

Вимоги:

1) Операційна система: Windows, macOS, Linux.

2) Мова програмування: Python.

3) Редактор коду: Visual Studio Code або інший.

4) Наявність бібліотек : psutil, Flask, smtplib, jwt, cryptography.

5) Доступ: локальний або мережевий доступ до серверів.

Методика інтеграції компонентів:

1) Встановлення системи:

Завантажити архів з програмним забезпеченням. Установити залежності за допомогою pip install requirements.txt. Перевірити наявність конфігураційного файлу config.json.

2) Налаштування параметрів:

Внести IP-адреси серверів, порогові значення, інтервали перевірок. Налаштувати email-оповіщення та графічний інтерфейс.

3) Запуск модулів:

Виконати скрипт: python app.py (відповідний модуль).

4) Доступ до інтерфейсу:

Перейти у браузері за відповідним посиланням.

5) Оцінка продуктивності:

Відбувається аналіз даних щодо перевантажень ресурсів та тестування на основі юніт-тестів.

Інструкція користувача:

Спочатку необхідно перейти до модулю документування для ознайомлення з алгоритмом дії:

```
================================================================================
Інструкція з використання системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів:

1. Запуск модуля конфігурація (cn.py) для встановлення значень (http://127.0.0.1:5000).

2. Запуск модуля моніторингу (mon.py) для збору метрик.

3. Запуск модуля аналізу даних (ad.py) для виконання аналізу метрик.

4. Запуск модуля перерозподілу ресурсів (ar.py) для очищення, уникнення перевантаження та вивільнення місця.

5. Запуск модуля оповіщення (no.py) для надсилання сповіщення на пошту у разі перевищення порогових значень.

6. Запуск модуля логування (lg.py) для перегляду логів (http://127.0.0.1:5000).

7. Запуск модуля безпеки (cr.py) для виконання процедури логіну (аутентифікації) та отримання JWT токена на основі якого відбудеться отримання доступу (http:/
/127.0.0.1:5000/login, http://127.0.0.1:5000/protected).

8. Запуск модуля тестування (tt.py)для перевірки функціонування програми.

-- Для вимкнення кожного з модулів потрібно натиснути комбінацію клавіш Ctrl+C --
-- Для модулів конфігурації, логування та безпеки необхідно перейти за посиланнями --
================================================================================
```

Рисунок 5.1 – Алгоритм дії

Для оновлення конфігурації необхідно перейти за посиланням після запуску модуля конфігурації:

Рисунок 5.2 – Оновлення конфігурації

Далі потрібно запустити відповідні модулі:

```
Starting monitoring... File: server_metrics.log
Metrics: {'datetime': '2025-05-09 08:51:41', 'cpu_usage': 12.1, 'memory_usage': 75.4, 'disk_usage': 67.1, 'network': {'bytes_sent': 18181440, 'bytes_recv': 5386426
1}}
Metrics: {'datetime': '2025-05-09 08:51:47', 'cpu_usage': 18.3, 'memory_usage': 75.5, 'disk_usage': 67.1, 'network': {'bytes_sent': 18184429, 'bytes_recv': 5386620
5}}
Metrics: {'datetime': '2025-05-09 08:51:53', 'cpu_usage': 4.6, 'memory_usage': 75.3, 'disk_usage': 67.1, 'network': {'bytes_sent': 18187855, 'bytes_recv': 53873162
}}
Monitoring stopped
```

Рисунок 5.3 – Моніторинг

```
Starting monitoring and analysis data...
Metrics: {'datetime': '2025-05-09 09:18:40', 'cpu_usage': 20.5, 'memory_usage': 73.4, 'disk_usage': 67.1, 'network': {'bytes_sent': 21368851, 'bytes_recv': 5947349
5}}
Data analysis result: {'overload': False, 'critical': False, 'desc': []}
Metrics: {'datetime': '2025-05-09 09:18:46', 'cpu_usage': 18.6, 'memory_usage': 73.3, 'disk_usage': 67.1, 'network': {'bytes_sent': 21387721, 'bytes_recv': 5949539
7}}
Data analysis result: {'overload': False, 'critical': False, 'desc': []}
Metrics: {'datetime': '2025-05-09 09:18:52', 'cpu_usage': 53.4, 'memory_usage': 75.9, 'disk_usage': 67.1, 'network': {'bytes_sent': 21564453, 'bytes_recv': 5952614
8}}
Data analysis result: {'overload': True, 'critical': False, 'desc': ['High memory usage: 75.9%']}
Monitoring and analysis data stopped
```

Рисунок 5.4 – Аналіз даних

```
Starting monitoring, analysis data and redistribution...
Metrics: {'datetime': '2025-05-09 10:34:12', 'cpu_usage': 18.1, 'memory_usage': 78.8, 'disk_usage': 67.1, 'network': {'bytes_sent': 38413571, 'bytes_recv': 3088208
86}}
Data analysis result: {'overload': True, 'critical': False, 'desc': ['High memory usage: 78.8%']}
Actions: ['Freeing memory...']
Metrics: {'datetime': '2025-05-09 10:34:19', 'cpu_usage': 9.0, 'memory_usage': 78.7, 'disk_usage': 67.1, 'network': {'bytes_sent': 38432142, 'bytes_recv': 30883274
2}}
Data analysis result: {'overload': True, 'critical': False, 'desc': ['High memory usage: 78.7%']}
Actions: ['Freeing memory...']
Metrics: {'datetime': '2025-05-09 10:34:25', 'cpu_usage': 10.3, 'memory_usage': 73.3, 'disk_usage': 67.1, 'network': {'bytes_sent': 38443935, 'bytes_recv': 3088443
48}}
Data analysis result: {'overload': False, 'critical': False, 'desc': []}
Monitoring, analysis data and redistribution stopped
```

Рисунок 5.5 – Перерозподіл ресурсів

Для модуля оповіщення необхідно вказати пошту та пароль для надсилання сповіщення:

```
7   G_EMAIL = 'testgmmli@gmail.com'
8   G_PASSWORD = 'vfnsuwubofupyipe'
```

```
Starting monitoring, analysis data, redistribution and notification...
Metrics: {'datetime': '2025-05-09 10:46:09', 'cpu_usage': 37.5, 'memory_usage': 79.5, 'disk_usage': 67.0, 'network': {'bytes_sent': 43176008, 'bytes_recv': 4117339
21}}
Data analysis result: {'overload': True, 'critical': False, 'desc': ['High memory usage: 79.5%']}
Actions: ['Freeing memory...']
Notification sent: Warning
Metrics: {'datetime': '2025-05-09 10:46:14', 'cpu_usage': 39.0, 'memory_usage': 79.5, 'disk_usage': 67.0, 'network': {'bytes_sent': 43202826, 'bytes_recv': 4117663
04}}
Data analysis result: {'overload': True, 'critical': False, 'desc': ['High memory usage: 79.5%']}
Actions: ['Freeing memory...']
Notification sent: Warning
Metrics: {'datetime': '2025-05-09 10:46:19', 'cpu_usage': 48.2, 'memory_usage': 78.6, 'disk_usage': 67.0, 'network': {'bytes_sent': 43209839, 'bytes_recv': 4117757
07}}
Data analysis result: {'overload': True, 'critical': False, 'desc': ['High memory usage: 78.6%']}
Actions: ['Freeing memory...']
Notification sent: Warning
Monitoring, analysis data, redistribution and notification stopped
```

Рисунок 5.6 – Вказання параметрів та оповіщення

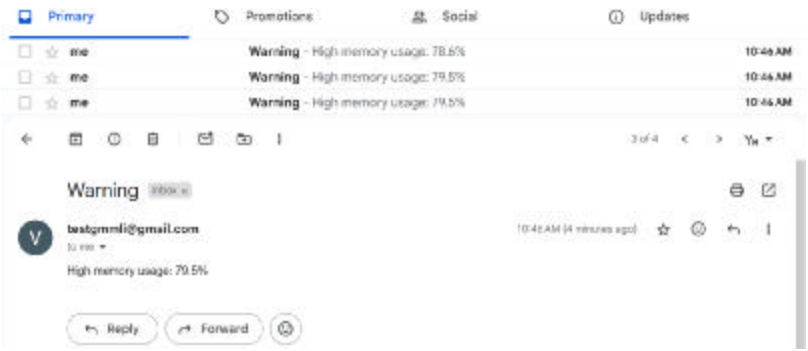

Рисунок 5.7 – Перелік та вміст повідомлень

Для перегляду логів необхідно перейти за посиланням після запуску модуля логування:

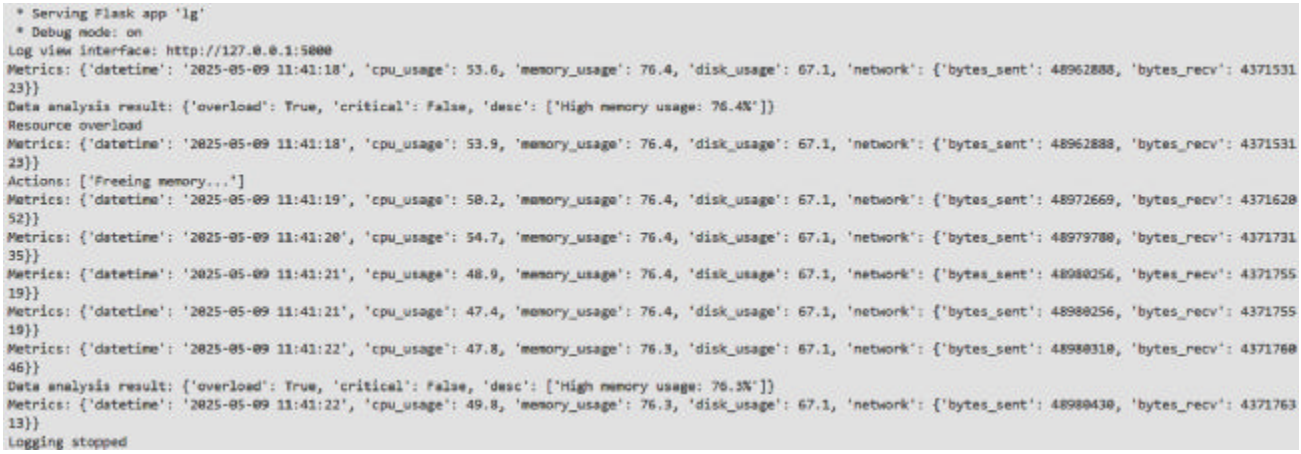

Рисунок 5.8 – Логування

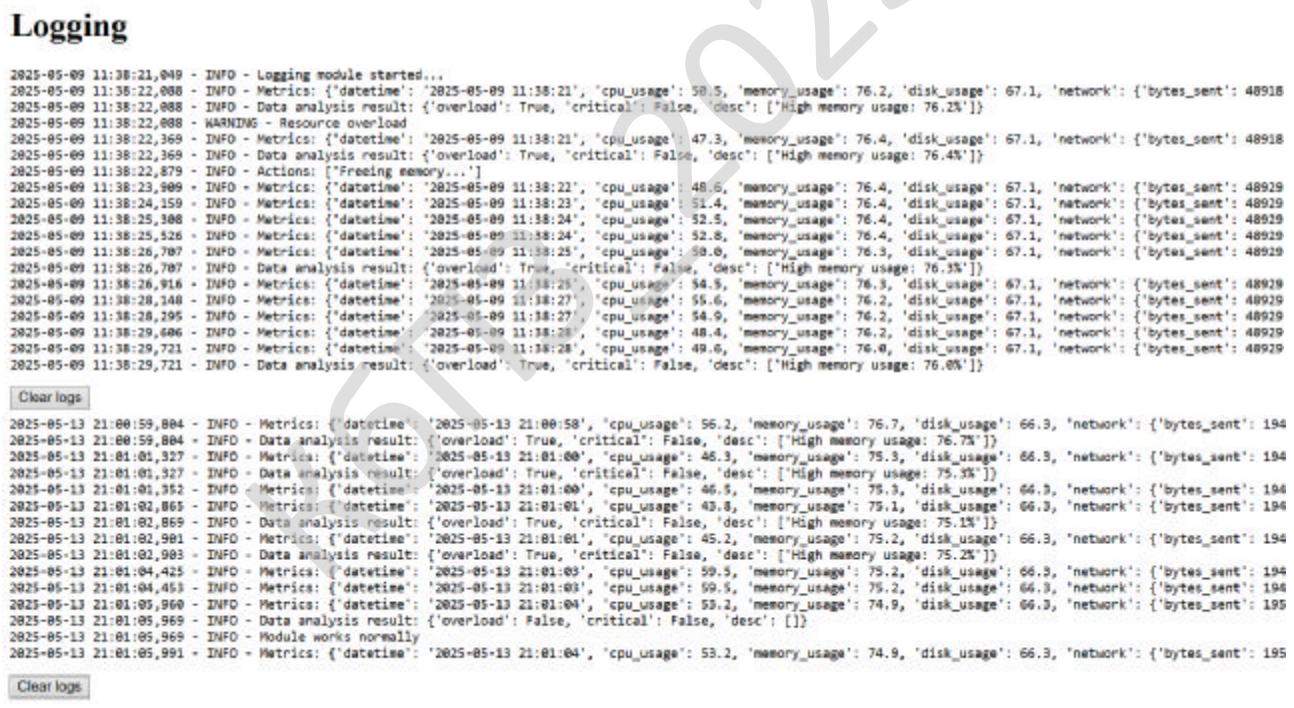

Рисунок 5.9 – Перегляд логів

Рисунок 5.10 – Очищення логів

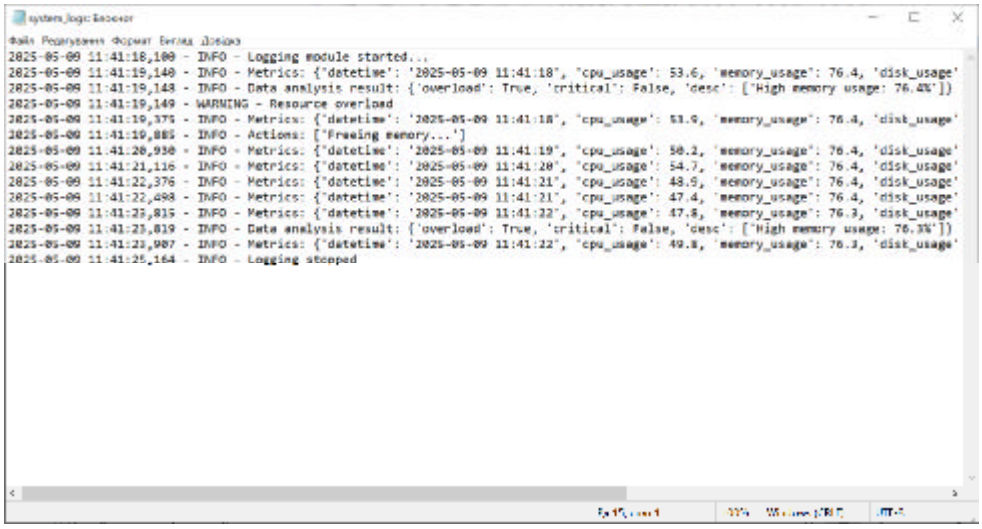

Рисунок 5.11 – Запис логів у файл

Після запуску модуля безпеки потрібно виконати процедуру логіну для отримання JWT токену і потім ввести його для отримання доступу:

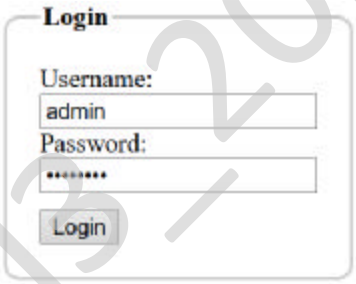

Рисунок 5.12 – Логін

Рисунок 5.13 – Отримання токену

Рисунок 5.14 – Введення токену

Access allowed for admin

Рисунок 5.15 – Надання доступу

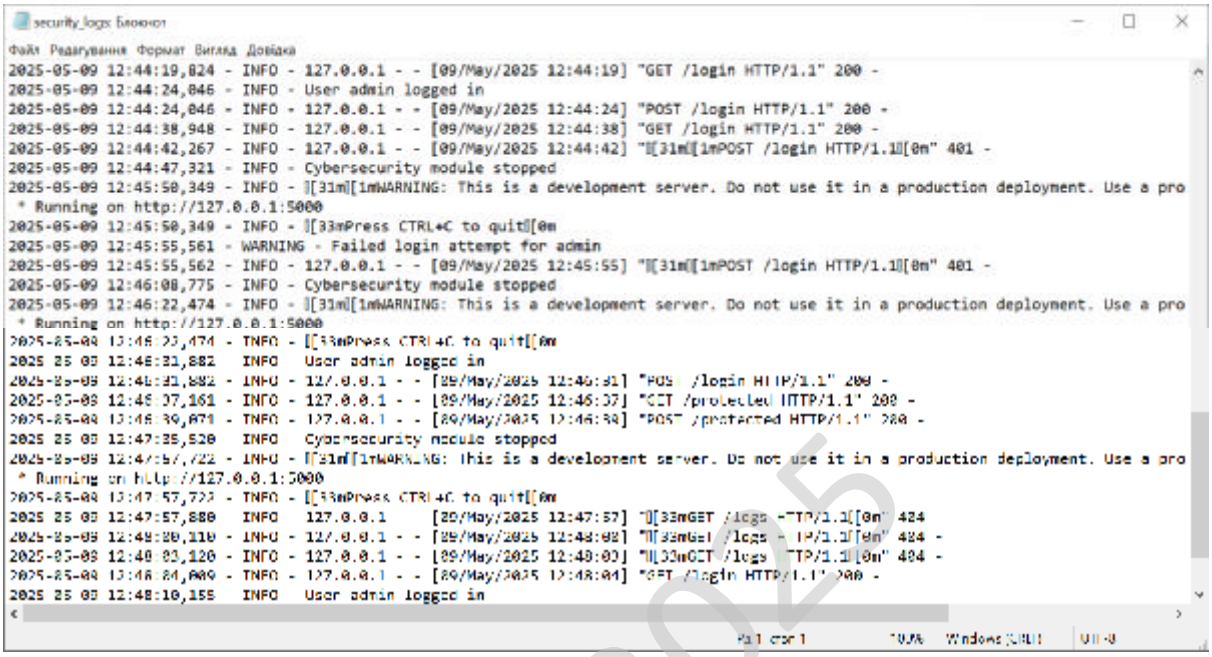

Рисунок 5.16 – Запис логів у файл

Після виконання всіх модулів виконується тестування модулів на основі юніт-тестів:

```
....
----------------------------------------------------------------------
Ran 4 tests in 1.051s

OK
```

Рисунок 5.17 – Тестування

Таким чином, кожен із модулів виконує відіграє важливу роль і виконує відповідні функції. Інтерфейс надає можливості для оновлення конфігурації, перегляду логів та забезпечення кібербезпеки. Правильність роботи програми можна перевірити за змінами метрик, логами подій та результатами виконаних дій у відповідь на перевантаження.

6 ОСНОВНІ ВИСНОВКИ

Таким чином, в процесі виконання курсової роботи були використані всі знання та вміння, отримані при вивченні дисциплін з програмування, інформаційної безпеки, комп'ютерних систем та мереж. Було розроблено та реалізовано програмне забезпечення системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації. Система відповідає сучасним вимогам до стабільності, інформативності, масштабованості та безпеки.

Також було покращено навички оформлення програмної документації.

В процесі розробки ПЗ були досягнуті наступні результати:

- реалізовано дев'ять функціональних модулів;
- створено інтерфейс на Flask;
- забезпечено можливість оновлення конфігурацій;
- впроваджено захищений обмін даними;
- протестовано систему;
- розроблено документацію.

Розроблена система дозволяє ефективно відстежувати навантаження на сервери, вчасно реагувати на критичні події, автоматизувати процес перерозподілу ресурсів та зменшити ризики втрати або пошкодження важливої інформації завдяки логуванню. Програмний продукт може бути адаптований для різних підприємств, організацій та розширений для підтримки хмарних сервісів і віддалених інтерфейсів.

Поставлені в роботі цілі та задачі виконані повністю. Розроблена система має значний потенціал для впровадження в інформаційні структури підприємств та подальшого вдосконалення з урахуванням сучасних загроз у сфері кібербезпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Погребняк А. В. Технології комп'ютерної безпеки : монографія. Рівне : МЕГУ, 2011. – 117 с.

2. С. Г. Семенов, А. О. Подорожняк, О. І. Баленко, С. Ю. Гавриленко Захист інформації в комп'ютерних системах та мережах : навч. посіб. Харків : НТУ «ХПІ», 2014. – 251 с.

3. J. R. Vacca Cybersecurity and IT infrastructure protection : 1st ed. Waltham : Elsevier, 2014. – 381 p.

4. Monitoring Tool Market Insights – Growth & Forecast 2024-2034 [Електронний ресурс]. – Режим доступу до ресурсу: https://www.futuremarketinsights.com/reports/monitoring-tool-market

5. Моніторинг Prometheus: варіанти використання, показники та найкращі практики [Електронний ресурс]. – Режим доступу до ресурсу: https://itedu.center/ua/blog/review/monitorynh-prometheus-varianty-vykorystannia-pokaznyky-ta-naikrashchi-praktyky

6. Prometheus & Grafana: що потрібно знати [Електронний ресурс]. – Режим доступу до ресурсу: https://itedu.center/ua/blog/articles/prometheus-and-grafana

7. Zabbix: Моніторинг мережевої безпеки [Електронний ресурс]. – Режим доступу до ресурсу: https://blog.silvery.ua/zabbix-network-security-monitoring

8. Zabbix: Що це і як ним користуватися [Електронний ресурс]. – Режим доступу до ресурсу: https://alexhost.com/uk/faq/zabbix-what-it-is-and-how-to-use-it

9. Топ 10 кращих програм для моніторингу мереж у 2025 [Електронний ресурс]. – Режим доступу до ресурсу: https://www.softinventive.com.ua/best-network-monitoring-tools

10. 10 кращих інструментів для системного адміністратора [Електронний ресурс]. – Режим доступу до ресурсу: https://surl.li/dkouxh

11. Nagios Log Server 4-Instance [Електронний ресурс]. – Режим доступу до ресурсу: https://allsoft.ua/p1771748676-nagios-log-server-4-instance.html?

12. Які недоліки Nagios? [Електронний ресурс]. – Режим доступу до ресурсу: https://shaft.uansver.cx.ua/maysternist/yaki-nedoliki-nagios.html

13. SolarWinds [Електронний ресурс]. – Режим доступу до ресурсу: https://oberig-it.com/solution_manf/solarwinds

14. SolarWinds Network Monitoring Software In-Depth Review [Електронний ресурс]. – Режим доступу до ресурсу: https://thectoclub.com/tools/solarwinds-review

15. Вибір між Prometheus, Datadog та New Relic [Електронний ресурс]. – Режим доступу до ресурсу: https://itedu.center/ua/blog/comparisons/vybir-mizh-prometheus-datadog-ta-new-relic/?

16. Що робить Datadog? [Електронний ресурс]. – Режим доступу до ресурсу: https://tvorchist.will.cx.ua/vzaiemodiya/shho-robit-datadog.html

17. 3 Straightforward Pros and Cons of Datadog for Log Analytic [Електронний ресурс]. – Режим доступу до ресурсу: https://www.chaossearch.io/blog/pros-cons-datadog-log-analytics

18. Icinga [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Icinga

19. Icinga pros and cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.peerspot.com/products/icinga-pros-and-cons

20. Netdata [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Netdata

21. Netdata Reviews & Product Details [Електронний ресурс]. – Режим доступу до ресурсу: https://www.g2.com/products/netdata/reviews

22. Checkmk [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Checkmk

23. Checkmk pros and cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.peerspot.com/products/checkmk-pros-and-cons

24. Splunk [Електронний ресурс]. – Режим доступу до ресурсу: https://www.mogroup.com.ua/?p=607

25. Splunk enterprise Reviews [Електронний ресурс]. – Режим доступу до ресурсу: https://www.joinsecret.com/splunk-enterprise/reviews

26 Завдання ELK Stack: логування та аналіз даних в DevOps-середовищах [Електронний ресурс]. – Режим доступу до ресурсу: https://itedu.center/ua/blog/review/elk-stack-tasks-logging-data-analysis/?

27. 5 ELK Stack Pros and Cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.chaossearch.io/blog/elk-stack-pros-and-cons

28. Paessler PRTG Network Monitor [Електронний ресурс]. – Режим доступу до ресурсу: https://allsoft.ua/ru/p1640873603-prtg-network-monitor.html?srsltid=AfmBOorAuQotreF97dMW2pJJTb4vzsB3gYaajg5pXqugs5OjRqLdoCw6

29. PRTG Network Monitor pros and cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.peerspot.com/products/prtg-network-monitor-pros-and-cons

30. Cisco AppDynamics – корпоративні програми під контролем [Електронний ресурс]. – Режим доступу до ресурсу: https://winncom.ua/blog/2018/12/06/cisco-appdynamics-korporativnye-prilozheniya-pod-kontrolem/news

31. Thousandeyes vs Appdynamics [Електронний ресурс]. – Режим доступу до ресурсу: https://www.netcomlearning.com/blog/thousandeyes-vs-appdynamics

32 OpenNMS [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/OpenNMS

33. The top 3 Advantages and Disadvantages to OpenNMS [Електронний ресурс]. – Режим доступу до ресурсу: https://surli.cc/msxbas

34. 16 найкращих інструментів хмарного моніторингу (2025) [Електронний ресурс]. – Режим доступу до ресурсу: https://www.guru99.com/uk/best-cloud-monitoring-tools.html

35. Site24x7 Reviews [Електронний ресурс]. – Режим доступу до ресурсу: https://www.softwareadvice.com/cloud-management/site24x7-profile/reviews

36. What is Sensu? A Complete Guide to Free Open Source Infrastructure, Applications, and Services Monitoring [Електронний ресурс]. – Режим доступу до ресурсу: https://dbsnoop.com/what-is-sensu-applications-services-monitoring

37. Sensu Reviews [Електронний ресурс]. – Режим доступу до ресурсу: https://www.softwareadvice.com/application-performance-manage/sensu-profile/reviews

38 Network Monitoring Software & Solutions - ManageEngine OpManager [Електронний ресурс]. – Режим доступу до ресурсу: https://www.manageengine.com/network-monitoring

39. ManageEngine OpManager Plus [Електронний ресурс]. – Режим доступу до ресурсу: https://www.g2.com/products/site24x7-manageengine-opmanager-plus/reviews?qs=pros-and-cons

40. 7 основних факторів, які слід врахувати при виборі програмного забезпечення для моніторингу мережі [Електронний ресурс]. – Режим доступу до ресурсу: https://progress.bakotech.com/ua/home/ua-7-key-considerations-to-make-before-choosing-a-network-monitoring-software-1

41. WhatsUp Gold pros and cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.peerspot.com/products/whatsup-gold-pros-and-cons

42. Вибір між Prometheus, Datadog та New Relic [Електронний ресурс]. – Режим доступу до ресурсу: https://itedu.center/ua/blog/comparisons/vybir-mizh-prometheus-datadog-ta-new-relic/?srsltid

43. New Relic Reviews [Електронний ресурс]. – Режим доступу до ресурсу: https://www.softwareadvice.com/devops/new-relic-one-profile/reviews

44. TrueSight Operations Management 11.x: Advanced Event Management [Електронний ресурс]. – Режим доступу до ресурсу: https://www.bmcsoftware.fr/education/courses/truesight-operations-management-11-x-advanced-event-management.html

45. BMC TrueSight Operations Management pros and cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.peerspot.com/products/bmc-truesight-operations-management-pros-and-cons

46. Zenoss [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Zenoss

47. Zenoss Cloud pros and cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.peerspot.com/products/zenoss-cloud-pros-and-cons

48. Centreon [Електронний ресурс]. – Режим доступу до ресурсу: https://endoflife.date/centreon

49. Centreon pros and cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.peerspot.com/products/centreon-pros-and-cons

50. LogicMonitor [Електронний ресурс]. – Режим доступу до ресурсу: https://www.morningdough.com/uk/ai-tools/logicmonitor

51. LogicMonitor pros and cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.peerspot.com/products/logicmonitor-pros-and-cons

52. EventSentry Overview [Електронний ресурс]. – Режим доступу до ресурсу: https://www.eventsentry.com/documentation/overview/html/overview.htm

53. EventSentry [Електронний ресурс]. – Режим доступу до ресурсу: https://www.g2.com/products/eventsentry/reviews?qs=pros-and-cons

54. Pandora FMS pros and cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.peerspot.com/products/pandora-fms-pros-and-cons

55. Cabot vs Sensu [Електронний ресурс]. – Режим доступу до ресурсу: https://stackshare.io/stackups/cabot-vs-sensu

56. CABOT SOFTWARE & APP [Електронний ресурс]. – Режим доступу до ресурсу: https://yachtica.com/our-products/cabot-software

57. Cabot - monitor and alert [Електронний ресурс]. – Режим доступу до ресурсу: https://www.cabotapp.com/use/services.html

58. Opsview pros and cons [Електронний ресурс]. – Режим доступу до ресурсу: https://www.peerspot.com/products/opsview-pros-and-cons

59. Мова програмування Пайтон [Електронний ресурс]. – Режим доступу до ресурсу: https://www.scribd.com/presentation/741930885

60. Python [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Python

61. Стандартна бібліотека Python [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.python.org/uk/3/library/index.html

62. Python: Незамінний інструмент у сучасному світі [Електронний ресурс]. – Режим доступу до ресурсу: https://platma.academy/python-nezaminnij-instrument-u-suchasnomu-sviti

63. Структурна схема [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Структурна_схема

64. Structural Diagrams | Unified Modeling Language(UML) [Електронний ресурс]. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/structural-diagrams-unified-modeling-languageuml

65. Функціональна схема [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Функціональна_схема

66. What is a Process Flow Diagram [Електронний ресурс]. – Режим доступу до ресурсу: https://www.lucidchart.com/pages/process-flow-diagrams

67. Flowchart [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Flowchart

68. Каплун В. А., Дмитришин О. В., Баришев Ю. В. Захист програмного забезпечення. Частина 2 : навч. посіб. Вінниця : ВНТУ, 2014. – 105 с.

Додаток А

Технічне завдання

ЗМІСТ

					ВКРБ-125.25.0005.00.00.ТЗ		
Вим.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Меркулов В.Р.			Програмне забезпечення системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації	Літ. Б	Аркуш 1 / Аркушів 6
Перев.		Дрєєв О.М.					
Н.контр.		Коваленко А.С.				ЦНТУ КБ-22МБ	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 51-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформції.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є відповідно до теми список літератури та існуючі аналоги.

5 Технічні вимоги

5.1 Вміст проекту

Складовими розробки є:

– аналіз існуючих систем моніторингу та кібербезпеки серверів на предмет їх відповідності вимогам до захищеності, стабільності та адаптивності;

– вибір і обґрунтування структури системи, засобів реалізації та мови програмування;

– розробка структури даних, взаємозв’язку між модулями;

– розробка програми, яка реалізує ряд модулів: моніторингу, аналізу, перерозподілу, оповіщення, логування, конфігурації, безпеки, тестування та документації.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформції;

– простий, інтуїтивно зрозумілий інтерфейс;

– цілісність даних.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно містити обмежень на версію драйверів, ОС, зміну параметрів та розширюваність модулів.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати об'єктно-орієнтовані, системні, програмні засоби розробки та підтримку облікових засобів системи.

5.5 Вимоги до надійності

Програмні модулі розроблені відповідно до всіх вимог, які стосуються викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів системи повинні задовольняти наступним умовам експлуатації:

– температура повітря: 18-27 градусів за Цельсієм;

– відносна вологість повітря до 80%;

– атмосферний тиск 100-107 кПа.

5.7 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і орієнтоване на сумісні з цією платформою пристрої, обладнання і прикладне програмне забезпечення.

5.8 Вимоги до інформаційної та програмної сумісності

Сумістність програмного забезпечення повинна бути забезпечена за рахунок його реалізації з використанням об'єктно-орієнтованого підходу, інтерфейсу, які працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена в вигляді опису структури даних, схем і описів алгоритмів, інструкції користувача, а також текстів вхідних модулів програмного забезпечення в відповідності з ЄСПД.

7 Перелік документів, які необхідно розробити

Структурна схема системи повна – 1 аркуш.
Функціональна схема системи – 1 аркуш.
Діаграма процесів – 1 аркуш.
Блок-схеми алгоритму роботи програми – 2 аркуша.
Пояснювальна записка – 64 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок-схем алгоритмів роботи програмного забезпечення компоненту.

8.4 Побудова схем взаємодії структур даних.

8.5 Створення прототипу ПЗ.

8.6 Відлагодження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю і приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 24.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 02.06.2025 р.

Додаток Б

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Дрєєв О.М.

Програмне забезпечення системи кібербезпеки моніторингу серверів з метою вчасного перерозподілу ресурсів для забезпечення цілісності інформації

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 20

Літера: РП

Кропивницький - 2025 року

Файл mon.py

```
import psutil
import time
import json
class Serversmonitoring:
    # ініціалізація
    def __init__(s, log_file='server_metrics.log', interval=5):
        s.log_file = log_file
        s.interval = interval
    # збір метрик
    def col_metr(s):
        metrics = {
            'datetime': time.strftime('%Y-%m-%d %H:%M:%S'),
            'cpu_usage': psutil.cpu_percent(interval=1),
            'memory_usage': psutil.virtual_memory().percent,
            'disk_usage': psutil.disk_usage('/').percent,
            'network': {
                'bytes_sent': psutil.net_io_counters().bytes_sent,
                'bytes_recv': psutil.net_io_counters().bytes_recv
            }
        }
        return metrics
    # логування метрик
    def log_metr(s, metrics):
        with open(s.log_file, 'a') as f:
            f.write(json.dumps(metrics) + '\n')
    # запуск моніторингу з інтервалом
    def start_mon(s):
        print("Starting monitoring... File:", s.log_file)
        try:
            while True:
                metrics = s.col_metr()
                s.log_metr(metrics)
                print("Metrics:", metrics)
                time.sleep(s.interval)
        except KeyboardInterrupt:
            print("Monitoring stopped")
if __name__ == '__main__':
    mon = Serversmonitoring(interval=5)
    mon.start_mon()
```

Файл ad.py

```
import time
from mon import Serversmonitoring
class Dataanalysis:
    # ініціалізація
    def __init__(s, thresholds=None):
        s.thresholds = thresholds or {
            'cpu': 80,
            'memory': 75,
            'disk': 85
        }
    # аналіз даних для визначення навантаженості
    def anldt(s, metrics):
        iss = {
            'overload': False,
            'critical': False,
            'desc': []
        }
        if metrics['cpu_usage'] >= s.thresholds['cpu']:
            iss['overload'] = True
             iss['desc'].append("High cpu usage: " + str(metrics['cpu_usage']) +
"%")
        if metrics['memory_usage'] >= s.thresholds['memory']:
            iss['overload'] = True
           iss['desc'].append("High memory usage: " + str(metrics['memory_usage'])
+ "%")
        if metrics['disk_usage'] >= s.thresholds['disk']:
            iss['overload'] = True
           iss['desc'].append("High disk usage: " + str(metrics['disk_usage']) +
"%")
        if metrics['cpu_usage'] >= s.thresholds['cpu'] and \
           metrics['memory_usage'] >= s.thresholds['memory'] and \
           metrics['disk_usage'] >= s.thresholds['disk']:
            iss['critical'] = True
        return iss
if __name__ == '__main__':
    mon = Serversmonitoring(interval=5)
    anser = Dataanalysis()
    # запуск аналізу даних на основі моніторингу
    print("Starting monitoring and analysis data...")
    try:
        while True:
            metrics = mon.col_metr()
            analysis_result = anser.anldt(metrics)
            print("Metrics:", metrics)
            print("Data analysis result:", analysis_result)
            time.sleep(mon.interval)
    except KeyboardInterrupt:
        print("Monitoring and analysis data stopped")
```

Файл ar.py

```
import psutil
import time
from ad import Dataanalysis
from mon import Serversmonitoring
class Resourcereallocation:
    # ініціалізація
    def __init__(s, thresholds=None):
        s.thresholds = thresholds or {
            'cpu': 80,
            'memory': 75,
            'disk': 85
        }
    # обробка процесора (від навантажень)
    def cpu_us(s):
        for process in psutil.process_iter(['pid', 'name', 'cpu_percent']):
            if process.info['cpu_percent'] > 50:
                psutil.Process(process.info['pid']).suspend()
                print("High cpu process stopped: ", process.info['name'], " (PID:
", str(process.info['pid']), ")")
    # обробка памяті (від навантажень)
    def mem_us(s):
        for process in psutil.process_iter(['pid', 'name', 'memory_info']):
            if process.info['memory_info'].rss > 900 * 1024 * 1024:
                psutil.Process(process.info['pid']).terminate()
                 print("High memory process stopped: ", process.info['name'], "
(PID: ", str(process.info['pid']), ")")
    # обробка диску (від навантажень)
    def disk_us(s):
        import os
        temp_dirs = ['/tmp', 'C:\\Windows\\Temp']
        for temp_dir in temp_dirs:
            if os.path.exists(temp_dir):
                for file in os.listdir(temp_dir):
                    file_p = os.path.join(temp_dir, file)
                    try:
                        os.remove(file_p)
                        print("Temp deleted:", file_p)
                    except Exception as e:
                        print("Cannot delete ", file_p, ": ", str(e))
    # забезпечення перерозподілу ресурсів на основі метрик
    def re_res(s, metrics):
        actions = []
        if metrics['cpu_usage'] > s.thresholds['cpu']:
            actions.append("Cpu redistribution...")
            s.cpu_us()
        if metrics['memory_usage'] > s.thresholds['memory']:
            actions.append("Freeing memory...")
            s.mem_us()
        if metrics['disk_usage'] > s.thresholds['disk']:
            actions.append("Disk cleaning...")
            s.disk_us()
        return actions
if __name__ == '__main__':
    mon = Serversmonitoring(interval=5)
    anser = Dataanalysis()
    redor = Resourcereallocation()
    # запуск модулів
    print("Starting monitoring, analysis data and redistribution...")
    try:
        while True:
            metrics = mon.col_metr()
            print("Metrics:", metrics)
            analysis_result = anser.anldt(metrics)
```

```
        print("Data analysis result:", analysis_result)
        if analysis_result['overload']:
            actions = redor.re_res(metrics)
            print("Actions:", actions)
        time.sleep(mon.interval)
except KeyboardInterrupt:
    print("Monitoring, analysis data and redistribution stopped")
```

Файл no.py

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from ad import Dataanalysis
from mon import Serversmonitoring
from ar import Resourcereallocation
# основні дані для взаємодії
G_EMAIL = 'testgmmli@gmail.com'
G_PASSWORD = 'vfnsuwubofupyipe'
# стан оповіщення
G_NOTIFICATION = {
    'warning': {
        'thval': {'cpu': 50, 'memory': 70, 'disk': 75},
        'subject': 'Warning',
        'rec': ['testgmmli@gmail.com']
    },
    'critical': {
        'thval': {'cpu': 85, 'memory': 80, 'disk': 90},
        'subject': 'Critical',
        'rec': ['testgmmli@gmail.com']
    }
}
# відправка оповіщення на пошту
def send_emnot(subject, message, rec):
    try:
        msg = MIMEMultipart()
        msg['From'] = G_EMAIL
        msg['To'] = ', '.join(rec)
        msg['Subject'] = subject
        msg.attach(MIMEText(message, 'plain'))
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login(G_EMAIL, G_PASSWORD)
        server.sendmail(G_EMAIL, rec, msg.as_string())
        server.quit()
        print(f"Notification sent: {subject}")
    except Exception as e:
        print(f"Failed to send notification: {e}")
# запуск модулів
def ano():
    mon = Serversmonitoring(interval=5)
    anser = Dataanalysis()
    redor = Resourcereallocation()
        print("Starting monitoring, analysis data, redistribution and
notification...")
    try:
        while True:
            metrics = mon.col_metr()
            analysis_result = anser.anldt(metrics)
            print("Metrics:", metrics)
            print("Data analysis result:", analysis_result)
            if analysis_result['overload']:
                actions = redor.re_res(metrics)
                print("Actions:", actions)
                not_st = 'critical' if analysis_result['critical'] else 'warning'
                inc_metrics = "\n".join(analysis_result['desc'])
                subject = G_NOTIFICATION[not_st]['subject']
                rec = G_NOTIFICATION[not_st]['rec']
                send_emnot(subject, inc_metrics, rec)
            mon.log_metr(metrics)
    except KeyboardInterrupt:
            print("Monitoring, analysis data, redistribution and notification
stopped")
```

```
if __name__ == '__main__':
    ano()
```

Файл lg.py

```
import logging
import threading
import signal
from flask import Flask, render_template_string, jsonify
from mon import Serversmonitoring
from ad import Dataanalysis
from ar import Resourcereallocation
log = logging.getLogger('werkzeug')
log.setLevel(logging.ERROR)
# формування системи логів (файлу)
logging.basicConfig(
    filename='system_logs.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)
stop_monitoring = threading.Event()
mon_act = False
# виконання та стан подій логів
def log_ev(st, message):
    with open('system_logs.log', 'r') as f:
        if message in f.read():
            return
    if st == 'info':
        logging.info(message)
    elif st == 'warning':
        logging.warning(message)
    elif st == 'error':
        logging.error(message)
    print(message)
app = Flask(__name__)
@app.route('/')
# відображення логів (Flask)
def disp_logs():
    try:
        with open('system_logs.log', 'r') as f:
            logs = f.readlines()
    except FileNotFoundError:
        logs = ['No logs']
    html_temp = """
    <html>
        <head>
            <title>Logging module</title>
            <script>
                function gLogs() {
                    fetch('/logs')
                    .then(response => response.json())
                    .then(data => {
                              document.getElementById('log-content').innerText =
data.logs;
                    });
                }
                function cLogs() {
                    fetch('/clear', {method: 'POST'})
                    .then(response => response.json())
                    .then(data => {
                        fetchLogs();
                    });
                }
                setInterval(gLogs, 3000);
                window.onload = gLogs;
            </script>
        </head>
        <body>
```

```
            <h1>Logging</h1>
            <pre id="log-content">Load logs...</pre>
            <button onclick="cLogs()">Clear logs</button>
        </body>
    </html>
    """
    return render_template_string(html_temp, logs=''.join(logs))
@app.route('/clear', methods=['POST'])
# очищення логів
def c_logs():
    open('system_logs.log', 'w').close()
    log_ev('info', 'Logs cleared')
    return jsonify({'message': 'Logs cleared'})
@app.route('/logs')
# отримання логів
def g_logs():
    try:
        with open('system_logs.log', 'r') as f:
            logs = f.read()
    except FileNotFoundError:
        logs = 'No logs'
    return jsonify({'logs': logs})
# запуск логування (усіх подій)
def start_mlog():
    global mon_act
    if mon_act:
        log_ev('info', 'Logging started')
        return
    mon_act = True
    mon = Serversmonitoring(interval=5)
    anser = Dataanalysis()
    redor = Resourcereallocation()
    log_ev('info', 'Logging module started...')
    print("Log view interface: http://127.0.0.1:5000")
    try:
        while not stop_monitoring.is_set():
            metrics = mon.col_metr()
            log_ev('info', "Metrics: {}".format(metrics))
            analysis_result = anser.anldt(metrics)
            log_ev('info', "Data analysis result: {}".format(analysis_result))
            if analysis_result['overload']:
                log_ev('warning', 'Resource overload')
                actions = redor.re_res(metrics)
                log_ev('info', "Actions: {}".format(actions))
            else:
                log_ev('info', 'Module works normally')
            mon.log_metr(metrics)
    except Exception as e:
        log_ev('error', "Error: {}".format(e))
# зупинка програми
def stop_prg(p1, p2):
    log_ev('info', 'Logging stopped')
    stop_monitoring.set()
    exit(0)
if __name__ == '__main__':
    signal.signal(signal.SIGINT, stop_prg)
    monitoring_thread = threading.Thread(target=start_mlog, daemon=True)
    monitoring_thread.start()
    app.run(debug=True)
```

Файл cn.py

```
import json
import threading
import time
import logging
import signal
from flask import Flask, request, render_template_string
# формування системи логів (файлу)
logging.basicConfig(
    filename = 'system_logs.log',
    level = logging.INFO,
    format = '%(asctime)s - %(levelname)s - %(message)s'
)
log = logging.getLogger('werkzeug')
log.setLevel(logging.ERROR)
# файл конфігурації
F_CON = 'config.json'
st_con = threading.Event()
class Configuration:
    # ініціалізація
    def __init__(s):
        s.config = s.l_conf()
        s.upd_conf()
    # завантаження конфігурації
    def l_conf(s):
        try:
            with open(F_CON, 'r') as f:
                data_con = json.load(f)
                logging.info("Configuration loaded")
                return data_con
        except (FileNotFoundError, json.JSONDecodeError) as e:
            logging.error(f"Error: {e}")
            return {}
    # отримання конфігурації (параметрів)
    def g_conf(s):
        return s.config
    # оновлення конфігурації
    def u_conf(s, new_con):
        try:
            with open(F_CON, 'w') as f:
                json.dump(new_con, f, indent=4)
            s.config = new_con
            logging.info("Configuration updated")
        except Exception as e:
            logging.error(f"Error: {e}")
    # динамічне оновлення конфігурації (без перезавантаження)
    def upd_conf(s):
        def monc():
            while not st_con.is_set():
                try:
                    with open(F_CON, 'r') as f:
                        data_con = json.load(f)
                        if data_con != s.config:
                            logging.info("Configuration updated dynamically")
                            s.config = data_con
                except Exception as e:
                    logging.error(f"Error: {e}")
                time.sleep(10)
        thread = threading.Thread(target=monc, daemon=True)
        thread.start()
con = Configuration()
app = Flask(__name__)
@app.route('/')
# відображення конфігурації (форми)
```

```
def disp_con():
    config = con.g_conf()
    html_temp = """
    <h1>Configuration control</h1>
    <form action="/update" method="post">
    <fieldset style="width: fit-content; padding: 20px; border: 2px solid #ccc;
border-radius: 10px;">
        <legend style="font-weight: bold;">Configuration</legend>
        <label>CPU:</label>
        <input type="number" name="cpu" value="{{ config['thresholds']['cpu'] }}"
style="margin-top: 10px;"><br>
        <label>Memory:</label>
                    <input     type="number"     name="memory"     value="{{
config['thresholds']['memory'] }}" style="margin-top: 10px;"><br>
        <label>Disk:</label>
        <input type="number" name="disk" value="{{ config['thresholds']['disk']
}}" style="margin-top: 10px;"><br>
        <label>Interval:</label>
        <input type="number" name="interval" value="{{ config['interval'] }}"
style="margin-top: 10px;"><br>
        <input type="submit" value="Update configuration" style="margin-top:
15px;">
    </fieldset>
    </form>
    """
    return render_template_string(html_temp, config=config)
@app.route('/update', methods=['POST'])
# дані, які оновлюються в конфігураційному файлі (після взаємодії з формою)
def u_conf():
    new_con = {
        "thresholds": {
            "cpu": int(request.form['cpu']),
            "memory": int(request.form['memory']),
            "disk": int(request.form['disk'])
        },
        "servers": con.g_conf()['servers'],
        "interval": int(request.form['interval'])
    }
    # остаточне оновлення конфігурації після натиснення кнопки
    con.u_conf(new_con)
    return '''
    <script>
        alert("Configuration updated");
        window.location.href = "/";
    </script>
    '''
# зупинка програми
def stop_prg(p1, p2):
    logging.info("Configuration stopped")
    print("Configuration stopped")
    st_con.set()
    exit(0)
# запуск програми
if __name__ == '__main__':
    signal.signal(signal.SIGINT, stop_prg)
    print("Configuration view interface: http://127.0.0.1:5000")
    app.run(debug=False)
```

Файл cr.py

```
import time
import signal
import hashlib
import logging
import threading
from flask import Flask, request, jsonify
from cryptography.fernet import Fernet
from collections import deque
import jwt
# формування системи логів (файлу)
logging.basicConfig(
    filename = 'security_logs.log',
    level = logging.INFO,
    format = '%(asctime)s - %(levelname)s - %(message)s'
)
inc_reqs = deque(maxlen=75)
ENC_KEY = Fernet.generate_key()
ciphr = Fernet(ENC_KEY)
JWT_SSK = "sseckey"
# БД для наявних користувачів
USER_DB = {
    "admin": hashlib.sha256("admin123".encode()).hexdigest(),
    "user": hashlib.sha256("user123".encode()).hexdigest()
}
app = Flask(__name__)
# виявлення ДДоС атаки
def dec_ddos():
    while True:
        cur_time = time.time()
        inc_reqs.append(cur_time)
        if len(inc_reqs) >= 55:
            tm_win = inc_reqs[-1] - inc_reqs[0]
            if tm_win < 15:
                logging.warning("DDoS")
                print("DDoS")

        time.sleep(1)
# шифрування даних
def encrypt_data(data):
    enc_data = ciphr.encrypt(data.encode())
    return enc_data
# дешифрування даних
def decrypt_data(enc_data):
    dec_data = ciphr.decrypt(enc_data).decode()
    return dec_data
# перевірка наявності файлу
def check_file_integrity(filename, expch):
    try:
        with open(filename, 'rb') as f:
            f_hs = hashlib.sha256(f.read()).hexdigest()
            return f_hs == expch
    except FileNotFoundError:
        logging.error("No file")
        return False
# генерація JWT токену для користувача
def generate_jwt(username):
    payld = {"user": username}
    token = jwt.encode(payld, JWT_SSK, algorithm="HS256")
    return token
# підтвердження токену
def verify_jwt(token):
    try:
        if token.startswith("Bearer "):
```

```
            token = token.split(" ")[1]
        payld = jwt.decode(token, JWT_SSK, algorithms=["HS256"])
        return payld["user"]
    except jwt.ExpiredSignatureError:
        return "Expired token"
    except jwt.InvalidTokenError:
        return "Invalid token"
@app.route('/login', methods=['GET', 'POST'])
# забезпечення логіну користувача для отримання токену (форма)
def login():
    if request.method == 'GET':
        return '''
        <form action="/login" method="post">
         <fieldset style="width: fit-content; padding: 20px; border: 2px solid
#ccc; border-radius: 10px;">
            <legend style="font-weight: bold;">Login</legend>
            <label>Username:</label><br>
            <input type="text" name="username"><br>
            <label>Password:</label><br>
            <input type="password" name="password"><br>
            <input type="submit" value="Login" style="margin-top: 10px;">
        </fieldset>
        </form>
        '''
    elif request.method == 'POST':
        data = request.form or request.json
        username = data.get('username')
        password = data.get('password')
                if    username    in    USER_DB    and    USER_DB[username]    ==
hashlib.sha256(password.encode()).hexdigest():
            token = generate_jwt(username)
            logging.info("User {} logged in".format(username))
            return f'''
            <h1>Login</h1>
            <p>Token:</p>
                        <textarea   readonly   style="width:   100%;   height:
100px;">{token}</textarea>
            <p>Token for protected resources</p>
            '''
        else:
            logging.warning("Failed login attempt for {}".format(username))
            return '<h1>Incorrect username or password</h1>', 401
@app.route('/protected', methods=['GET', 'POST'])
# забезпечення перевірки токену та отримання доступу на його основі (форма)
def protected():
    if request.method == 'GET':
        return '''
        <h1>Access</h1>
        <form action="/protected" method="post">
            <label>Enter token:</label><br>
            <input type="text" name="token" style="width: 100%;"><br><br>
            <input type="submit" value="Check access">
        </form>
        '''
    elif request.method == 'POST':
        token = request.form.get("token")
        if not token:
            return '<h1>Token required</h1>', 401
        if token.startswith("Bearer "):
            token = token.split(" ")[1]
        user = verify_jwt(token)
        if user == "Invalid token" or user == "Expired token":
            return f'<h1>{user}</h1>', 401
        return f'<h1>Access allowed for {user}</h1>'
# зупинка програми
```

```
def stop_program(p1, p2):
    logging.info("Cybersecurity module stopped")
    print("Cybersecurity module stopped")
    exit(0)
# запуск програми
if __name__ == '__main__':
    signal.signal(signal.SIGINT, stop_program)
    print("Starting cybersecurity module...")
    print("Login: http://127.0.0.1:5000/login")
    print("Entering and validating a JWT token: http://127.0.0.1:5000/protected")
    threading.Thread(target=dec_ddos, daemon=True).start()
    app.run(debug=False)
```

Файл tt.py

```
import unittest
import unittest
from mon import Serversmonitoring
from ad import Dataanalysis
from cr import encrypt_data, decrypt_data, generate_jwt, verify_jwt
class Testmodules(unittest.TestCase):
    # ініціалізація обєктів для подальшого використання
    def setUp(s):
        s.mon = Serversmonitoring()
        s.anser = Dataanalysis()
    # тестування отримання метрик
    def test_cmetr(s):
        mtr = s.mon.col_metr()
        s.assertIsInstance(mtr, dict)
        s.assertIn('cpu_usage', mtr)
        s.assertIn('memory_usage', mtr)
        s.assertIn('disk_usage', mtr)
        s.assertIn('network', mtr)
    # тестування аналізу даних
    def test_anl(s):
        t_dt = {'cpu_usage': 80, 'memory_usage': 75, 'disk_usage': 85, 'network':
{}}
        res = s.anser.anldt(t_dt)
        s.assertIsInstance(res, dict)
        s.assertTrue(res['overload'])
        s.assertTrue(res['critical'])
    # тестування шифрування та дешифрування
    def test_encr(s):
        test_text = "Data"
        enc = encrypt_data(test_text)
        dec= decrypt_data(enc)
        s.assertEqual(test_text, dec)
    # тестування отримання та зчитування токену JWT
    def test_jwt(s):
        token = generate_jwt("admin")
        user = verify_jwt(token)
        s.assertEqual(user, "admin")
if __name__ == "__main__":
    unittest.main()
```

Файл dc.py

```
# покроковий алгоритм дій
def disp_ins():
    act_alg = """
    ================================================================================
============
    Інструкція з використання системи кібербезпеки моніторингу серверів з метою
вчасного перерозподілу ресурсів:

       1. Запуск модуля конфігурація (cn.py) для встановлення значень
(http://127.0.0.1:5000).

    2. Запуск модуля моніторингу (mon.py) для збору метрик.

    3. Запуск модуля аналізу даних (ad.py) для виконання аналізу метрик.

     4. Запуск модуля перерозподілу ресурсів (ar.py) для очищення, уникнення
перевантаження та звільнення місця.

    5. Запуск модуля оповіщення (no.py) для надсилання сповіщення на пошту у разі
перевищення порогових значень.

        6. Запуск модуля логування (lg.py) для перегляду логів
(http://127.0.0.1:5000).

       7. Запуск модуля безпеки (cr.py) для виконання процедури логіну
(аутентифікації) та отримання JWT токена на основні якого відбудеться отримання
доступу (http://127.0.0.1:5000/login, http://127.0.0.1:5000/protected).

    8. Запуск модуля тестування (tt.py)для перевірки функціонування програми.

    -- Для вимкнення кожного з модулів потрібно натиснути комбінацію клавіш Ctrl+C
--
     -- Для модулів конфігурації, логування та безпеки необхідно перейти за
посиланнями --
    ================================================================================
============
    """
    print(act_alg)
if __name__ == "__main__":
    disp_ins()
```