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 адміністрування доступу до мережних пристроїв”

КбПЗ_2025

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Бойко Ю.О.
«____» _____ 2025 р.

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

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

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

Бойко Юлії Олександрівні
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи адміністрування доступу до мережних пристроїв*

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

затверджені наказом вищого навчального закладу № 47-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи адміністрування доступу до мережних пристроїв*

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*

КПІ-2025

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	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	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Марченко К.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Бойко Ю.О.
(прізвище та ініціали)

КбПЗ_2025

АНОТАЦІЯ

Бойко Ю.О. Програмне забезпечення системи адміністрування доступу до мережних пристроїв. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи адміністрування доступу до мережних пристроїв.

Метою розробки є програмне забезпечення системи адміністрування доступу до мережних пристроїв.

Результат роботи – програмна реалізація системи адміністрування доступу до мережних пристроїв.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, адміністрування доступу, мережні пристрої

КПЗ – 2025

ABSTRACT

Boyko Yu.O. Software for the network device access administration system. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the network device access administration system.

The purpose of the development is software for the network device access administration system.

The result of the work is the software implementation of the network device access administration system.

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 convenient user interface has been developed. Instructions for working with the software are provided.

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

The program was developed in the Python environment.

Keywords: computer engineering, access administration, network devices

КБПЗ_2025

ЗМІСТ

КОПЗ_2025

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

EOM	–	електронно-обчислювальна машина
ATM	–	Asynchronous Transfer Mode – асинхронний спосіб передачі даних
BER	–	Bit Error Rate – імовірність ушкодження одного біта
CBWFQ	–	class based weighted fair queueing
DiffServ	–	Differentiated Services – механізм який залежно від вимог до якості обслуговування записує в IP заголовки пакетів спеціальні мітки
DSCP	–	DiffServ CoSde Point
ICMP	–	Internet Control Message Protocol – міжмережний протокол управляючих повідомлень
ISP	–	Internet Service Provider – провайдер
LLQ	–	low latency queueing
MPLS-TE	–	Multiprotocol Label Switching Traffic Engineering
PQ	–	priority queueing
RIO	–	RED with Input Output
RTT	–	Round Trip Time – час між відправкою запиту та отриманням відповіді
STM	–	Synchronous Transfer Mode – синхронний спосіб передачі даних
QoSS	–	Quality of Service – якість обслуговування
WFQ	–	Weighted Fair Queuing – механізм планування пакетних потоків даних
WRED	–	Weighted ranDoSm early detection – взвішане значення довжини черги, у якості фактора, визначаючого ймовірність відкидання пакета

КПЗ – 2025

ВСТУП

Актуальність теми. Навіть п'ятихвилинний простій мережі може привести до значних збитків, тому для запобігання непередбачених ситуацій необхідно застосовувати комплексний і структурований підхід до керування мережею, що передбачає в числі іншого виконання проактивних дій.

Завдання керування комп'ютерною мережею є невід'ємним, а в багатьох випадках і головному обов'язковим для мережного інженера або адміністратора. На перший погляд, що складного в цій роботі? З погляду рядового співробітника вона виглядає в такий спосіб: одержавши повідомлення від користувачів мережі про ті або інші несправності (наприклад, недоступний Інтернет, не підключається телефон до мережі, «гальмує» RDP), мережний інженер усуває їх. Такі роботи можуть бути охарактеризовані як реактивна підтримка.

На жаль, для обслуговування великої мережі реактивний підхід найчастіше виявляється недостатнім. У якийсь момент кількість повідомлень про несправності починає збільшуватися лавиноподібно, що може викликати відмову критичних мережних сервісів. Оскільки комп'ютерна мережа є бізнес-критичним інструментом для будь-якого сучасного підприємства, навіть п'ятихвилинної простої мережі може привести до значних збитків, тому мережний інженер повинен виключити можливість виникнення подібних ситуацій.

Для запобігання непередбаченої відмови варто застосовувати комплексний і структурований підхід до керування мережею, що передбачає в числі іншого виконання проактивних дій.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи адміністрування доступу до мережних пристроїв.

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

КПЗ-2025

– Огляд існуючих систем адміністрування доступу до мережних пристроїв.

– Дослідження системи адміністрування доступу до мережних пристроїв.

– Програмна реалізація системи адміністрування доступу до мережних пристроїв.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі адміністрування доступу до мережних пристроїв.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи адміністрування доступу до мережних пристроїв, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ_2025

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Зважаючи на те, що кіберзагрози постійно розвиваються, безпека вашої мережі полягає не лише в надійних паролях або брандмауерах, але й у тому, що потрібні люди мають доступ до потрібних ресурсів у потрібний час. Розуміння та впровадження ефективного керування доступом до мережі є наріжним каменем захисту цінних даних і підтримки ефективності роботи.

У цій роботі ми досліджуємо основи контролю та керування доступом до мережі, включаючи його основні функції, переваги та найкращі рішення на ринку.

Контроль доступу до мережі проти керування доступом до мережі

Контроль доступу до мережі (NAC) – це рішення безпеки, яке забезпечує застосування політик для контролю доступу пристроїв до мережі. Це гарантує, що лише авторизовані пристрої та користувачі можуть підключатися до мережевих ресурсів і що ці пристрої відповідають політикам безпеки перед отриманням доступу. Він зосереджений на контролі доступу до мережевих ресурсів, мережевих вузлів, таких як маршрутизатори, комутатори або брандмауери.

З іншого боку, керування доступом до мережі є більш широким поняттям, яке охоплює загальну стратегію, політику та технології, що використовуються для контролю та керування доступом до мережі. Він включає NAC як компонент, але також включає інші аспекти керування мережею. Основні функції NAM включають розробку політики та її застосування, надання та скасування доступу, а також моніторинг і звітність. NAM також тісно інтегровано з управлінням ідентифікацією та адмініструванням (IGA) , забезпечуючи належне узгодження прав доступу з політикою організації та вимогами відповідності.

КбПЗ–2025

1.2 Область застосування

Насамперед виділимо основні складові завдання керування мережею й спробуємо розібратися, яку роль у цьому процесі грає кожний з них.

1.Забезпечення доступу до мережних пристроїв для керування ними. Завдання забезпечення доступу до мережних пристроїв нестандартні, особливо у випадку більших територіально розподілених мережних інфраструктур.

2. Моніторинг мережної інфраструктури.

Як показує досвід інженерів, моніторинг, виконуваний у рамках описаних вище завдань, є необхідною й достатньою умовою для того, щоб мережна інфраструктура працювала на належному рівні. При акуратному дотриманні запропонованих (і, у загальному-те, нескладних) пунктів, мережний інженер зможе впоратися практично з будь-якою мережною проблемою в стислий термін. А багато мережних неполадок можуть бути усунуті проактивно.

Якщо враховані не всі запропоновані пункти, то рано або пізно виявиться, що для рішення чергової проблеми не вистачає інформації й необхідно негайно додавати відсутні засоби моніторингу, що зажадає додаткового часу. Крім того, навіть після установки цих засобів мережний інженер не зможе вірогідно інтерпретувати нові дані, адже в нього не буде інформації про те, як мережа працювали раніше і які показники вважати нормальними, а які аномальними.

3. Заміна застарілого або встаткування, що вийшло з ладу. Із часом надійність мережних пристроїв значно знижується: застаріле обладнання піддане новим видам атак, тому й мережна інфраструктура в цілому стає уразливою. Крім того, з певного часу застаріле обладнання перестає відповідати постійно зростаючим вимогам до функціональності й продуктивності. Таким чином, інженер повинен мати вичерпну інформацію про моделі використовуваних мережних пристроїв і при необхідності проводити заміну встаткування на більше сучасне.

КОПЗ_2025

4. Відновлення програмного забезпечення мережних пристроїв. Виробники мережного встаткування постійно розробляють нові версії операційних систем. Як правило, у них усуваються виявлені уразливості, виправляються помилки (баги) у коді, збільшується продуктивність, а також додаються нові функції.

5. Резервне копіювання конфігурацій мережних пристроїв. При відмові пристрою з високою часткою ймовірності губиться опис його конфігурації. пристрій, Що Вийшов з ладу, необхідно замінити на аналогічне – з такими ж налаштуваннями. Безумовно, це завдання значно спрощується, якщо є резервна копія конфігурації.

На корисність наявності такої копії вказує ще один приклад. Мережний інженер вносить зміни в конфігурацію мережного пристрою, після чого перестає працювати частина сервісів (можливо, бізнес-критичних). Така ситуація може відбутися як через неправильну дію (людський фактор), так і з вини виробника ПЗ (проявляється баг). Проте працездатність сервісу повинна бути відновлена в стислий термін. При наявності резервної копії конфігурації інженер може швидко здійснити процедуру відкоту, а потім виявити помилку в новій конфігурації.

6. Підтримка документації мережної інфраструктури. Як правило, первісна документація на мережну інфраструктуру створюється на етапах проектування й впровадження, а в процесі експлуатації вносяться численні зміни в налаштування пристроїв, у топологію мережі й т.д. Відсутність актуальної документації утрудняє як підтримку її роботи, так і усунення несправностей, особливо якщо обслуговуванням займаються відразу кілька людей.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи адміністрування доступу до мережних пристроїв, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КОПЗ-2025

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Контроль доступу до мережі має вирішальне значення для безпечної та сумісної IT-екосистеми. Рішення NAC контролюють доступ шляхом автентифікації користувачів і пристроїв і оцінки їх відповідності політикам безпеки, перш ніж дозволити доступ до мережі. Коли пристрій намагається підключитися до мережі, NAC визначає, який рівень доступу (якщо такий є) слід надати на основі таких критеріїв, як роль користувача, тип пристрою, місце розташування запиту та стан безпеки пристрою.

Навіщо потрібен контроль доступу до мережі?

Надійний контроль доступу до мережі є важливим для будь-якої сучасної стратегії кібербезпеки та дотримання нормативних вимог. Однією з причин є збільшення кількості та різноманітності пристроїв, яким сьогодні потрібен доступ до мережі. З появою Інтернету речей (IoT) кількість пристроїв, які мають доступ до мереж, різко зросла. Кожен пристрій, від простого офісного принтера до складних серверів, є потенційною точкою входу для зовнішніх зловмисників і внутрішніх загроз, випадкових чи навмисних. Крім того, без NAC сегментація мережі неможлива.

Величезний обсяг цих пристроїв робить непрактичним, якщо не неможливим, контролювати доступ до мережі вручну, тому комплексна стратегія NAC, яка використовує автоматизацію, є життєво необхідною. Можливості рішень NAC розвинулися, щоб відповідати зростаючій складності IT-середовища, включаючи зростаючу різноманітність пристроїв користувачів, підвищення вимог до віддаленого та мобільного доступу та поширення пристроїв IoT.

КПЗ - 2025

Крім того, системи NAC допомагають забезпечити дотримання нормативних стандартів, гарантуючи, що пристрої, які отримують доступ до мережі, відповідають законам про захист даних і конфіденційність.

Основні функції контролю доступу до мережі

Комплексна стратегія контролю доступу до мережі передбачає такі функції NAC:

Механізм політики

Рішення NAC зазвичай використовують центральний механізм політики, який використовує політики контролю доступу. Ці політики безпеки можуть включати такі критерії, як ролі користувачів, типи пристроїв, місцезнаходження, час доби та стан безпеки пристрою. Вони можуть бути досить детальними, вказуючи дозволи на доступ на рівні користувача, пристрою чи програми.

Оцінка кінцевої точки

NAC оцінює стан безпеки пристроїв, які намагаються підключитися до мережі, перш ніж дозволити доступ. Це може включати перевірку наявності антивірусного програмного забезпечення, виправлень операційної системи, налаштувань брандмауера та інших конфігурацій безпеки. Пристрої, які не відповідають вимогам безпеки, можуть бути поміщені на карантин або мати обмежений доступ, доки вони не відповідатимуть.

Точки контролю мережі

Точки контролю мережі – це комутатори, маршрутизатори, точки бездротового доступу, шлюзи VPN, брандмауери та інші точки входу в мережу. Вони забезпечують виконання політик NAC на основі рішень механізму політики.

Моніторинг мережі, профілювання та звітування

Системи NAC постійно відстежують мережевий трафік і активність, щоб ідентифікувати та профілювати пристрої, що підключаються до мережі, на предмет спроб несанкціонованого доступу, порушень політики та інцидентів безпеки. Це дає адміністраторам доступ до мережі, поведінку користувачів і стан

КБПЗ-2025

відповідності в реальному часі. Детальні журнали та звіти допомагають у криміналістичному аналізі, аудиті відповідності та реагуванні на інциденти безпеки. Поведінковий аналіз і методи зняття відбитків пальців допомагають ідентифікувати пристрої на основі їхніх моделей зв'язку в мережі та характеристик.

Механізми автентифікації та авторизації

Після підключення до мережі користувачі та пристрої проходять аутентифікацію, щоб підтвердити свою особу. Після автентифікації NAC визначає відповідні дозволи доступу на основі особи користувача, характеристик пристрою та інших контекстних факторів. Рішення про авторизацію приймаються в режимі реального часу відповідно до політик контролю доступу, визначених механізмом політики.

Методи автентифікації користувача

Приклади методів автентифікації користувачів:

– Користувачі надають унікальне ім'я користувача та відповідний пароль для автентифікації.

– Користувачі повинні надати кілька форм автентифікації, наприклад пароль у поєднанні з тимчасовим кодом, надісланим на їхній мобільний пристрій.

– Щоб підтвердити свою особу, користувачі надають цифрові сертифікати, видані надійним центром сертифікації.

– Користувачі ідентифікуються за допомогою таких фізіологічних характеристик, як відбитки пальців, сканування райдужної оболонки ока або розпізнавання обличчя.

Методи ідентифікації пристрою

Методи ідентифікації пристроїв включають наступне:

– MAC-адреси унікально ідентифікують мережеві інтерфейси на пристроях.

– Адреси Інтернет-протоколу (IP) унікально ідентифікують пристрої в мережі.

КБПЗ – 2025

– Пристрої надають цифрові сертифікати для автентифікації в мережі.

– Такі характеристики пристрою, як операційна система, тип пристрою та постачальник, використовуються для ідентифікації пристроїв.

Доступ для гостей і BYOD

NAC надає механізми для керування доступом для гостьових користувачів і сценаріїв із використанням власного пристрою (BYOD). Користувачам-гостьам може бути надано обмежений доступ до певних ресурсів на основі визначених політик. Пристрої BYOD проходять профільування пристрою та оцінку безпеки, перш ніж отримати доступ, із дозволами доступу, які визначаються на основі статусу відповідності пристрою.

Динамічний контроль доступу

NAC може динамічно регулювати дозволи доступу на основі змінних умов у мережевому середовищі. Наприклад, якщо стан безпеки пристрою погіршується через те, що його антивірусне програмне забезпечення вимкнено, NAC може обмежити його доступ, доки проблему безпеки не буде усунено.

Карантин і санація

У випадках, коли пристрої не проходять оцінку безпеки або порушують політики доступу, NAC може помістити їх у карантин, щоб запобігти подальшому впливу мережі. Пристрої на карантині поміщаються в обмежений сегмент мережі з обмеженим доступом, доки вони не відповідатимуть політикам безпеки. Механізми виправлення допомагають привести невідповідні пристрої у відповідність, надаючи інструкції або автоматичні виправлення.

Інтеграція NAC з існуючою мережевою інфраструктурою

Щоб контролювати доступ, рішення NAC зазвичай інтегруються з такими мережевими компонентами:

– **Комутатори та маршрутизатори** – рішення NAC часто інтегруються з мережевими комутаторами та маршрутизаторами для забезпечення виконання політик контролю доступу на межі мережі. Інтеграція з комутаторами дозволяє NAC динамічно призначати VLAN або застосовувати списки контролю доступу

КПЗ – 2025

(ACL) на основі статусу автентифікації користувача та пристрою. Маршрутизатори можуть застосовувати політики доступу для VPN-з'єднань віддаленого доступу, щоб забезпечити підключення до мережі лише автентифікованим користувачам.

– **Бездротові точки доступу (WAP)** – рішення NAC інтегруються з бездротовою інфраструктурою для забезпечення виконання політик контролю доступу для мереж Wi-Fi. WAP використовують такі протоколи, як IEEE 802.1X, для автентифікації користувачів і пристроїв перед наданням доступу до бездротової мережі. Інтеграція з NAC дозволяє WAP застосовувати політики доступу на основі ідентифікації користувача, типу пристрою та стану безпеки.

– **Брандмауери** – рішення NAC можуть інтегруватися з брандмауерами для забезпечення політики контролю доступу для трафіку, що входить і виходить з мережі, а також сегментації мережі всередині. Брандмауери можуть використовувати інформацію про автентифікацію користувачів і пристроїв із системи NAC для застосування детальних правил контролю доступу на основі ролей користувачів, типів пристроїв та інших контекстних факторів.

– **Системи керування ідентифікацією та доступом (IAM)** – рішення NAC інтегруються з такими системами керування ідентифікацією, як Active Directory і LDAP, щоб оптимізувати процеси автентифікації та авторизації користувачів і забезпечити узгодженість у всій організації.

– **Системи безпеки та керування подіями (SIEM)** – рішення NAC інтегруються з системами SIEM для забезпечення централізованого моніторингу та звітування про події доступу до мережі. Інтеграція з SIEM дозволяє NAC співвідносити події контролю доступу з іншими подіями та сповіщеннями безпеки, покращуючи видимість потенційних загроз безпеці.

– **Рішення безпеки кінцевих точок** – рішення NAC можуть інтегруватися з рішеннями безпеки кінцевих точок, такими як антивірусне програмне забезпечення, системи виявлення та реагування на кінцеві точки (EDR) і платформи керування мобільними пристроями (MDM). Інтеграція з цими

КОПЗ-2025

рішеннями дозволяє NAC оцінювати пристрої та застосовувати політики контролю доступу на основі їх статусу безпеки.

Переваги впровадження NAC

Впровадження контролю доступу до мережі пропонує кілька переваг для організацій, які прагнуть підвищити кібербезпеку та ефективно керувати доступом до мережі. NAC допомагає зменшити ризики, пов'язані з доступом некерованих пристроїв до корпоративних ресурсів. Нижче наведено деякі основні переваги:

– **Покращений рівень безпеки**. Перевіряючи особистість користувачів і оцінюючи пристрої перед наданням доступу, NAC допомагає зменшити ризик витоку даних і простою.

– **Покращена відповідність** – контролюючи та перевіряючи доступ до мережі, NAC допомагає організаціям досягати, підтримувати та демонструвати відповідність GDPR , HIPAA, PCI DSS та іншим нормам.

– **Швидше виявлення загроз і реагування** – Завдяки безперервному моніторингу мережевої активності NAC допомагає IT-командам виявляти потенційні загрози безпеці в режимі реального часу. NAC також може автоматизувати реагування на загрози, помістивши в карантин пристрої, які становлять загрозу безпеці, щоб запобігти їм доступ до конфіденційних мережевих ресурсів, доки проблема не буде усунена.

– **Краща видимість і контроль** – NAC надає адміністраторам можливість бачити в режимі реального часу доступ до мережі, типи пристроїв і стан відповідності. Така видимість дозволяє їм ефективніше виявляти інциденти безпеки та реагувати на них, а також завчасно вдосконалювати політики безпеки. Це також дозволяє сегментувати мережу та контролювати її.

– **Спрощені процеси реєстрації** – NAC допомагає оптимізувати процес реєстрації для нових користувачів і пристроїв шляхом автоматизації автентифікації користувачів і реєстрації пристроїв. Це зменшує адміністративне навантаження на IT-персонал і забезпечує безпечну та ефективну реєстрацію

КПЗ – 2025

нових користувачів і пристроїв.

– **Підтримка ініціатив BYOD** – NAC сприяє безпечним ініціативам BYOD, застосовуючи контроль доступу та політики безпеки для персональних пристроїв, які підключаються до корпоративної мережі.

– **Оптимізована продуктивність мережі**. Шляхом динамічного налаштування дозволів доступу на основі ролей користувачів, типів пристроїв і стану безпеки NAC допомагає оптимізувати продуктивність мережі та використання пропускної здатності. Це також гарантує, що мережеві ресурси розподіляються ефективно та встановлюються пріоритети на основі бізнес-вимог.

– **Централізоване керування політикою** – NAC надає централізовану платформу для визначення, керування та застосування політик контролю доступу в усій організації. Це спрощує керування політикою, забезпечує послідовність і зменшує ризик неправильної конфігурації та конфліктів політик.

Типи рішень НАК

Існує три основних типи рішень NAC: апаратні, програмні та хмарні. Кожен тип має переваги та міркування, і організації повинні оцінити свої конкретні вимоги, мережеву архітектуру та цілі безпеки, перш ніж вибрати найбільш прийнятне рішення.

Апаратні системи NAC

Апаратні рішення NAC зазвичай являють собою пристрої, спеціально створені для контролю доступу та забезпечення безпеки. Вони можуть включати спеціалізовані апаратні компоненти для обробки завдань автентифікації, авторизації та виконання політики.

Спеціальні пристрої пропонують передбачувану продуктивність і надійність, що робить їх придатними для критично важливих мережевих середовищ. Вони часто розгортаються вбудовано в мережеву інфраструктуру, що дозволяє їм перехоплювати та перевіряти мережевий трафік у режимі реального часу.

Крім того, апаратні системи NAC розроблені для масштабування для

КБПЗ_2025

підтримки великої кількості користувачів і пристроїв у розподілених мережевих середовищах. Ці системи пропонують високу пропускну здатність і низьку затримку, забезпечуючи мінімальний вплив на продуктивність мережі навіть за великих навантажень трафіком. Вони можуть підтримувати кластеризацію або конфігурації високої доступності для резервування та відмовостійкості.

Програмні рішення NAC

Програмні рішення для контролю доступу до мережі використовують програмні додатки або віртуальні пристрої. Ці рішення пропонують гнучкість, масштабованість і легкість розгортання. Вони дозволяють організаціям розгортати функціональні можливості NAC за потреби та масштабувати ресурси на основі мінливих вимог мережі, що робить їх придатними для різних мережевих середовищ. Програмні рішення NAC можна розгортати локально в приватних або гібридних хмарних середовищах.

Програмні рішення NAC можуть підтримувати як агентні, так і безагентні моделі розгортання:

– **NAC на основі агента** – Рішення NAC на основі агента вимагають встановлення програмних агентів на пристроях користувачів. Ці агенти збирають інформацію про стан безпеки пристрою та застосовують політики контролю доступу на основі цієї інформації. Рішення NAC на основі агентів пропонують детальний контроль і видимість окремих пристроїв, але можуть вимагати додаткових витрат для розгортання та керування агентом.

– **NAC без агентів** – Рішення NAC без агентів не потребують встановлення програмних агентів на пристроях користувачів. Натомість вони використовують мережеві механізми, такі як автентифікація IEEE 802.1X, відбитки DHCP або аналіз мережевого трафіку, щоб оцінити стан безпеки пристрою та застосувати політики контролю доступу. Безагентні рішення NAC забезпечують легкість розгортання та масштабованість, але можуть мати обмеження щодо видимості та контролю порівняно з рішеннями на основі агентів.

КОПІЯ 2025

Хмарні рішення NAC

Хмарні рішення NAC побудовані на власних хмарних архітектурах і використовують хмарну інфраструктуру для масштабованості, надійності та продуктивності. Ці пропозиції програмного забезпечення як послуги (SaaS) усувають необхідність локального розгортання апаратного чи програмного забезпечення та пропонують розгортання без дотику, що дозволяє організаціям швидко та легко розгортати функції контролю доступу. Ресурси можна збільшувати або зменшувати залежно від попиту, забезпечуючи оптимальну продуктивність і економічну ефективність.

Хмарні послуги NAC також забезпечують глобальне охоплення та доступність, дозволяючи організаціям застосовувати політики контролю доступу для користувачів і пристроїв незалежно від місця розташування. Користувачі можуть безпечно підключатися до корпоративних ресурсів з будь-якої точки світу, включаючи віддалені офіси, філії та мобільні пристрої. Хмарні служби NAC інтегруються з компонентами хмарної та локальної мережевої інфраструктури, забезпечуючи безперебійне підключення та контроль доступу в гібридних середовищах.

Критичні міркування щодо розгортання рішень NAC

Впровадження рішення NAC вимагає ретельного планування, налаштування та тестування. Нижче наведено ключові елементи успіху.

Оцінка та планування

Визначте та задокументуйте цілі розгортання NAC. Ретельно оцініть інфраструктуру мережі, включаючи апаратне забезпечення, програмне забезпечення та топологію мережі. Визначте обсяг розгортання, включаючи кількість користувачів, пристроїв і сегментів мережі, які мають бути охоплені. Визначте конкретні випадки використання та сценарії, коли NAC забезпечить цінність, наприклад захист віддаленого доступу, застосування політик BYOD або забезпечення відповідності нормативним стандартам.

КПЗ_2025

Бюджет і ресурси

Оцініть бюджет і ресурси, необхідні для розгортання NAC, включаючи витрати на апаратне/програмне забезпечення, послуги впровадження, навчання та поточне обслуговування. Розглянемо загальну вартість володіння (TCO) і рентабельність інвестицій (ROI), щоб виправдати розгортання.

Визначення політики

Визначте політики контролю доступу на основі організаційних вимог, найкращих практик безпеки та нормативних актів. Враховуйте ролі користувачів, типи пристроїв, місцезнаходження, час доби та стан безпеки. Визначте правила політики для автентифікації користувачів, профілювання пристроїв, керування доступом на основі ролей і примусових дій (наприклад, карантин, виправлення). Обов'язково задокументуйте політики контролю доступу в централізованому сховищі політик для зручного використання та керування.

Вибір рішення NAC

Оцініть рішення NAC на основі їхніх функцій, можливостей інтеграції та сумісності з існуючою мережевою інфраструктурою та потенційних можливостей розширення мережі за ємністю (пристроєм і користувачем) або технологією (не дивлячись на мережевих постачальників). Також враховуйте модель розгортання (апаратна, програмна або хмарна), підтримку постачальника та економічну ефективність.

Архітектура розгортання

Розробіть архітектуру розгортання для NAC, враховуючи топологію мережі, моделі трафіку, а також вимоги безпеки та відповідності. Визначте, чи будуть точки застосування NAC розгорнуті в мережі чи поза смугою, і чи будуть вони розгорнуті в точках доступу до мережі, усередині чи в обох.

Інтеграція з мережевою інфраструктурою

Визначте, як інтегрувати рішення NAC з існуючими компонентами мережевої інфраструктури, включаючи комутатори, маршрутизатори, брандмауери, сервери автентифікації, системи IAM і рішення безпеки кінцевих

КбПЗ-2025

точок. Крім того, плануйте інтегрувати NAC із рішеннями мережевої безпеки, такими як система виявлення вторгнень (IDS), система запобігання вторгненням (IPS) і платформа SIEM.

Профілі користувачів і пристроїв

Налаштуйте політики контролю доступу в рішенні NAC на основі визначених правил політики та вимог. Визначте ролі користувачів, типи пристроїв і вимоги до автентифікації на основі організаційних потреб і політик безпеки.

Тестування та валідація

Проведіть ретельне тестування розгортання NAC у контрольованому середовищі, щоб підтвердити його функціональність, продуктивність і безпеку: протестуйте політики контролю доступу, механізми автентифікації, профілювання пристрою, оцінку безпеки кінцевої точки та можливості застосування політики. Усуньте будь-які проблеми, виявлені під час тестування, перед розгортанням NAC у робочому середовищі.

Розгортання та розгортання

Розгортайте NAC поетапно, починаючи з пілотного розгортання в невеликому контрольованому середовищі. На основі результатів пілотного розгортання поступово розширюйте розгортання на додаткові сегменти мережі, користувачів і пристрої.

Моніторинг і технічне обслуговування

Регулярно контролюйте розгортання NAC, щоб забезпечити постійну відповідність політикам контролю доступу, вимогам безпеки та цілям продуктивності. Використовуйте можливості моніторингу та звітності рішення NAC для моніторингу активності доступу до мережі, сповіщень безпеки та порушень політики.

Крім того, необхідно виконувати регулярні завдання з обслуговування, такі як оновлення програмного забезпечення, виправлення та зміни конфігурації, щоб забезпечити постійну ефективність і безпеку розгортання NAC.

КОПЗ – 2025

Обізнаність і навчання користувачів

Проведіть навчання користувачів і IT-персоналу щодо мети та переваг розгортання NAC. Навчіть їх дотримуватись політики контролю доступу та заходів безпеки, які застосовуються рішенням NAC.

Проблеми впровадження керування доступом до мережі

Готуючись до впровадження контролю доступу до мережі, обов'язково враховуйте наступні проблеми, з якими часто стикаються організації:

– **Складність мережевої інфраструктури**. Сучасні мережеві середовища стають дедалі складнішими, охоплюючи різні пристрої, платформи та варіанти підключення. Управління доступом у цьому неоднорідному середовищі може бути складним завданням, оскільки вимагає досвіду захисту різних систем і технологій.

– **Різноманітні типи користувачів і пристроїв**. Сьогодні організації мають різноманітні групи користувачів, включаючи співробітників, підрядників, гостей і партнерів, кожен із різними вимогами до доступу. Подібним чином типи пристроїв вийшли за межі традиційних робочих станцій і серверів і включають пристрої IoT і масив кінцевих точок BYOD. Керування доступом для цих різноманітних користувачів і пристроїв, зберігаючи безпеку, є серйозною проблемою.

– **Віддалений і мобільний доступ**. Із зростанням віддаленої роботи та мобільних комп'ютерів забезпечення безпечного доступу до корпоративних ресурсів з будь-якого місця та в будь-який час стало надзвичайно важливим. Для керування доступом для віддалених і мобільних користувачів потрібні надійні механізми автентифікації, шифрування та заходи безпеки кінцевих точок, щоб зменшити ризик витоку даних і простою.

– **Вимоги до відповідності** – керування доступом до мережі має відповідати положенням суворих нормативних актів, таких як GDPR, HIPAA та PCI DSS. Забезпечення відповідності цим нормам, дотримуючись балансу між безпекою та зручністю використання, може бути складним завданням, особливо в

КПЗ-2025

галузях із жорстким регулюванням.

– **Взаємодія з користувачем і продуктивність**. Незважаючи на те, що безпека має першочергове значення, впровадження надто обмежувального контролю доступу може призвести до розчарування та неефективності користувача. Організаціям часто важко збалансувати безпеку та зручність використання.

– **Масштабованість і продуктивність**. Масштабованість і продуктивність є критично важливими факторами, особливо для великих організацій із великою базою користувачів і мережевою інфраструктурою. Рішення NAC повинні масштабуватися, щоб відповідати зростанню та коливанню навантаження користувачів, зберігаючи при цьому оптимальну продуктивність і швидкість реагування.

– **Нові технології та загрози**. Швидкий технологічний прогрес, як-от хмарні обчислення, Інтернет речей, штучний інтелект (AI) і машинне навчання (ML), створює проблеми з керуванням доступом і створює загрози безпеці. Рішення NAC повинні адаптуватися до цих нових технологій і забезпечувати надійний захист від нових кіберзагроз.

– **Проблеми інтеграції**. Рішення NAC мають бездоганно інтегруватися з існуючими компонентами мережевої інфраструктури, системами керування ідентифікацією, засобами безпеки та іншими IT-системами. Досягнення сумісності та плавної інтеграції може бути складним, вимагаючи координації між постачальниками та технологіями.

– **Необхідність постійного моніторингу та адаптації**. Керування доступом до мережі є постійним процесом, який потребує постійного моніторингу, аналізу та адаптації до нових загроз і бізнес-вимог. Щоб не відставати, потрібна постійна пильність і інвестиції в навчання як IT-команд, так і бізнес-користувачів.

– **Обмеження ресурсів**. Впровадження NAC потребує значних ресурсів, зокрема часу, бюджету та кваліфікованого персоналу. Обмежені ресурси можуть

КПЗ_2025

перешкоджати процесу розгортання та впливати на ефективність впровадження NAC.

– **Опір змінам** – опір змінам з боку різних зацікавлених сторін може стати значною перешкодою для впровадження NAC. Бізнес-користувачі можуть чинити опір новим методам автентифікації чи політикам контролю доступу, тоді як IT-персонал може неохоче застосовувати незнайомі технології чи робочі процеси. Крім того, вище керівництво може не надати достатньої підтримки ініціативам NAC, оскільки вони не повністю усвідомлюють необхідність змін.

– **Застарілі системи** – застарілі системи або застаріла мережева інфраструктура можуть не повністю підтримувати вимоги NAC, створюючи проблеми сумісності та обмежуючи ефективність впровадження NAC. Для подолання цих перешкод може знадобитися оновлення або заміна застарілих систем.

Основні технології мережевої структури

Оскільки бізнес продовжує розвиватися, попит на гнучкі, масштабовані та безпечні мережеві рішення спонукав розвиток передових мережевих технологій. Ключовими серед цих технологій є наступні:

– **Програмно-визначена мережа (SDN)** відокремлює площину керування від площини даних у мережевому обладнанні. Централізувавши мережевий інтелект у програмному контролері, SDN забезпечує покращену програмованість, автоматизацію та гнучкість. Це дозволяє мережевим адміністраторам динамічно керувати мережевими ресурсами та оптимізувати потік трафіку, що сприяє підвищенню ефективності та зниженню операційних витрат.

– **Віртуалізація мережевих функцій (NFV)** перетворює традиційні мережеві функції, такі як брандмауери та балансувальники навантаження, у віртуалізовані служби, які працюють на стандартному обладнанні. Ця віртуалізація зменшує залежність від пропрієтарного обладнання, забезпечує більш гнучке розгортання послуг і полегшує масштабування мережевих функцій відповідно до мінливих вимог. NFV є невід'ємною частиною сучасних мережевих

КОПЗ_2025

інфраструктур, пропонуючи більш гнучку та економічно ефективну альтернативу традиційним мережевим пристроям.

– **Мікросегментація** ділить мережу на сегменти, кожен з яких має власний набір політик безпеки. На відміну від традиційної сегментації, яка ізолює великі сегменти мережі, мікросегментація дозволяє точніше контролювати трафік зі сходу на захід у центрі обробки даних. Цей підхід мінімізує поверхню атаки та підвищує безпеку, гарантуючи, що навіть якщо один сегмент скомпрометовано, злом буде локалізовано.

– **Сегментація мережі** передбачає поділ великої мережі на менші окремі сегменти для підвищення продуктивності та безпеки. Завдяки ізоляції різних частин мережі сегментація зменшує ризик широкомасштабних атак і дозволяє ефективніше керувати трафіком. Це також допомагає забезпечити контроль доступу, гарантуючи, що конфіденційні дані залишаються захищеними та доступними лише авторизованим користувачам.

Ці передові технології мережевої структури спільно сприяють створенню більш безпечного, ефективного та масштабованого мережевого середовища. Вони дають змогу організаціям краще керувати своїми ресурсами, захищати конфіденційні дані та швидко реагувати на зміни потреб бізнесу.

Тенденції та нові технології, що впливають на NAC

Нижче наведено деякі нові тенденції та технології, які вплинуть на NAC:

– **Нульова довіра** – модель безпеки Zero Trust швидко набирає популярності серед організацій. Він не надає довіри за замовчуванням, вимагаючи суворої перевірки особи та мінімального контролю привілеїв для кожного користувача та пристрою, які намагаються підключитися до мережі. NAC відіграє вирішальну роль у забезпеченні дотримання принципів нульової довіри , постійно перевіряючи ідентифікацію користувачів і пристроїв і коригуючи дозволи доступу на основі контекстних факторів.

– **Штучний інтелект і машинне навчання** – технології AI і ML інтегруються в рішення NAC для покращення можливостей виявлення загроз,

КОПЗ-2025

виявлення аномалій і аналізу поведінки. Рішення NAC на базі штучного інтелекту можуть визначати підозрілу активність, виявляти нові загрози та автоматично адаптувати засоби контролю доступу в режимі реального часу, щоб зменшити ризики безпеки, відповідності та безперервності бізнесу.

– **Інтеграція безпеки кінцевих точок** – рішення NAC все більше інтегруються з технологіями безпеки кінцевих точок, такими як виявлення кінцевих точок і реагування (EDR), антивірус і платформи захисту кінцевих точок (EPP), щоб покращити видимість, застосувати політики безпеки та реагувати на інциденти безпеки, пов'язані з кінцевими точками. Інтеграція з рішеннями безпеки кінцевих точок дозволяє NAC оцінювати стан безпеки пристроїв і впроваджувати засоби контролю доступу на основі рівнів ризику.

– **Безпека пристроїв IoT і OT** – Поширення пристроїв IoT і операційних технологій (OT) створює нові проблеми безпеки для керування доступом до мережі. Рішення NAC розвиваються для підтримки унікальних вимог середовищ IoT і OT, надаючи можливості виявлення пристроїв, профілювання та контролю доступу для захисту цих кінцевих точок і запобігання неавторизованому доступу.

– **Акцент на взаємодію з користувачем** – такі технології, як контекстна автентифікація, адаптивні політики доступу та портали самообслуговування, дозволять організаціям забезпечити плавний і безперешкодний доступ до мережевих ресурсів без шкоди для безпеки. Контроль доступу до мережі має важливе значення для захисту від кіберзагроз і досягнення відповідності нормам і галузевим стандартам. Основні аспекти NAC включають отримання повної видимості всіх пристроїв і користувачів, які підключаються до мережі, і розробку політик, які захищають безпеку без шкоди для взаємодії з користувачем. Організаціям також необхідно проводити часті навчальні сесії, щоб підвищити обізнаність користувачів щодо безпеки. Вибір рішення NAC із надійними функціями інтеграції дає змогу використовувати всі можливості стека безпеки, покращує виявлення загроз і реагування на них, а також оптимізує операції безпеки.

КБПЗ-2025

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

- у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;
- гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

КПЗ-2025

– розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;

– Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;

– у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив'язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп'ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для

КОПЗ_2025

веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

КБПЗ_2025

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи адміністрування доступу до мережних пристроїв.

В процесі розробки випускної кваліфікаційної роботи за першим

(бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

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

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

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

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

КОПЗ_2025

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Забезпечення доступу до мережних пристроїв для керування ними

При організації керування мережними пристроями необхідно вибрати схему керування, а також забезпечити безпечну можливість підключення до них з метою налаштування.

Класичні типи керування. Можна виділити кілька типів керування мережним устаткуванням. До найбільше часто описуваних методів відносяться керування по основній мережі (in-band) і по зовнішньому каналі (out-of-band). Крім класичного підходу, існують і альтернативні варіанти керування мережним устаткуванням. Наприклад, на ринку представлені готові рішення, де керування пристроями походить із хмари. Звичайно, такий метод можна було б віднести до одному із двох уже названих, але в силу його гібрідності й специфіки має сенс виділити його в окремий клас. Крім того, не варто забувати, що величезну популярність набирає концепція SDN (програмно конфігуруємі мережі). Отже, давайте розберемося, що собою представляє кожний тип.

Перший (in-band) припускає передачу трафіку керування встаткуванням (Telnet, SSH, HTTPs та ін.) і трафіку моніторингу (Syslog, SNMP, Netflow та ін.) через ті ж фізичні канали й порти на мережному встаткуванні, через які передається й звичайний користувальницький трафік. Інакше кажучи, та сама мережа забезпечує передачу всіх даних. Безумовно, при налаштуванні встаткування необхідно на логічному рівні сегментувати трафік керування й інший трафік. Це можна зробити за допомогою віртуальних мереж (VLAN), списків доступу (ACL), міжмережних екранів і іншого. Але, як було відзначено, фізична інфраструктура залишається тої ж.

КПЗ-2025

Основний плюс такого рішення – простота, але, як це звичайно буває, за простоту доводиться платити. Справа в тому, що у випадку збоїти в мережі (наприклад, через великого паразитний трафіку), доступ до пристроїв може бути перекритий, у результаті чого утрудняються їхня діагностика й усунення неполадок. Щоб цього не відбулося, на мережному встаткуванні необхідно настроїти різні рівні якості обслуговування трафіку (QoS). Трафіку керування й моніторингу варто надати пріоритет, виділивши мінімально необхідну пропускну здатність. Відповідним чином промартільний, він буде передаватися, навіть якщо мережа перевантажена. Однак даний підхід не дає 100-процентної гарантії доступності й ускладнює налаштування встаткування. Крім того, є ймовірність того, що з якоїсь причини пристрій автоматично заблокує порт і віддалений доступ до нього пропаде (наприклад, комутатор Cisco може перевести порт у стан err-disabled). Зрештою, бувають випадки, коли помилково блокується порт, через який іде трафік керування, і тоді губиться доступ до самого пристрою.

Другий тип керування (out-of-band) припускає передачу трафіку керування по окремих фізичних каналах зв'язку, для чого будується друга мережа, а на кожному мережному пристрої виділяється окремий порт для підключення до неї. Звичайно встановлюється й окремий комутатор, до якого підключається вся інфраструктура керування (машина адміністратора, сервер Syslog, колектор Netflow та інше). У такому випадку адміністратор мережі практично завжди буде мати доступ до мережного встаткування, навіть якщо основна мережа повністю вийде з ладу.

Безумовно, основним недоліком є необхідність використання окремого встаткування й виконання додаткових налаштувань. Крім того, при подібній організації керування завжди потрібні окремі канали зв'язку. Коли встаткування перебуває в одній серверній, завжди можна знайти додаткові комутаційні шнури, але якщо пристрої розміщені в різних частинах будинку або на територіально розподілених площадках, питання одержання виділених каналів стає критичним. Як це часто буває, додаткові мідні або оптичні траси не передбачені споконвічно,

КОПЗ-2025

прокласти їх досить проблематично. Крім того, на мережному встаткуванні повинні бути додаткові порти для підключення до мережі керування, але найчастіше в маршрутизаторах є всього два фізичних інтерфейси, і звичайно вони вже зайняті.

Що стосується територіально розподіленої мережі, то організувати керування по зовнішньому каналі у віддаленому офісі, особливо якщо його топологія досить проста, виявляється складно й дорого. Тому в такій ситуації застосовується гібридний варіант керування: по зовнішньому каналу для центрального офісу й по основній мережі для віддалених приміщень. Точно така ж схема може бути використана, коли мережа складається з декількох сегментів, розташованих на різних поверхах будинку, а додаткові міжповерхові з'єднання відсутні.

Хотілося б відзначити ще один варіант керування мережею, що більшою мірою можна віднести до зовнішнього керування, – використання консольного сервера, до якого підключається кожний мережний пристрій. Безсумнівна перевага даного варіанта полягає в тому, що, навіть якщо пристрій не змогла коректно завантажитися, доступ до нього буде забезпечений. Але використовувати тільки цей варіант заважають кілька нюансів. Забігаючи вперед, хотілося б відзначити, що консольний сервер може й повинен використовуватися як додатковий елемент в обох схемах керування.

Основні обмеження полягають у тому, що до консольного сервера зручно підключати ті пристрої, які перебувають на невеликому видаленні, тобто в одній серверній кімнаті з ним. Крім того, консольне підключення непридатне для моніторингу мережного встаткування. Ще один нюанс зв'язаний зі швидкістю передачі даних: вивід великої кількості інформації може зайняти досить тривалий час – у нашій практиці були випадки півгодинного очікування.

Крім консольного сервера, як альтернативний варіант можна використовувати спеціально виділений ноутбук з консольним кабелем. Цей мінімальний комплект для підключення до мережного встаткування по консолі

КПЗ – 2025

дозволяє при необхідності швидко дійти до пристрою й підключитися до нього. На жаль, найчастіше в потрібний момент під рукою не виявляється ноутбука або консольного кабелю з переходником.

Альтернативні типи керування

Як уже говорилося, керування із хмари являє собою до певної міри гібридний спосіб. Одним із прикладів реалізації керування мережним устаткуванням із хмари є концепція, застосована в лінійці продуктів Cisco Meraki. Всі пристрої цієї марки (а в неї входять і комутатори, і пристрою безпеки, і рішення по побудові бездротової мережі) після установки автоматично підключаються до хмари Cisco. Керування ними здійснюється через хмарний портал. Головним недоліком даної схеми є те, що при втраті зв'язку із хмарою управляти пристроями неможливо. Це істотно підвищує вимоги до надійності й кількості інтернет-каналів. Такий тип керування поки не придбав популярності, але через поширення хмарних технологій чекати залишилося недовго.

Концепція програмно конфігуруемих мереж бурхливо розвивається. Вона припускає повне відділення функцій керування пристроями й контролю трафіку від функцій передачі даних. Інакше кажучи, за керування всіма мережними пристроями й логікові контролю за трафіком (протоколи маршрутизації, службові протоколи, VLAN) буде відповідати якийсь централізований програмний пристрій (контролер), а мережне встаткування – займатися тільки передачею трафіку. З одного боку, переваги підходу в наявності: зручне керування всією мережею й дуже гнучка функціональність (додаткові функції реалізуються програмно). З іншого боку, мережні пристрої з підтримкою даної технології тільки починають з'являтися і їхньої можливості поки невеликі. Наскільки перспективний такий підхід, стане ясно в найближчі роки.

Коректне й безпечне налаштування мережного встаткування для керування

В ідеалі налаштування мережного встаткування для цілей керування й моніторингу виконується відповідно до рекомендацій виробника: паролі повинні

КПЗ – 2025

мати достатній ступінь надійності, для віддаленого підключення рекомендуються протоколи SSH і HTTPs, доступ до пристрою необхідно обмежити. Список можна продовжити й далі: коректне налаштування кожного протоколу керування, рекомендації із забезпечення якості обслуговування (QoS) і зняттю телеметрії з устаткування й т.д.

Опис налаштування всіх цих параметрів займає не одну сторінку, і їхній переказ навряд чи цікавий. Однак наскільки все це потрібно? Будь-який фахівець, звичайно ж, підтвердить виправданість таких дій. Тому вірніше сформулювати питання по-іншому: де та золота середина між достатнім рівнем безпечної керованості пристроями й потрібною глибиною моніторингу й складністю налаштування мережного встаткування? Адже, напевно, нема рації налаштовувати весь спектр функцій при будь-якому зручному випадку. Принаймні ми так не робимо. Безумовно, не варто впадати й у ту крайність, коли на встаткуванні навіть не міняються ім'я й пароль, установлені за замовчуванням, адже доступ до такого пристрою відкритий з будь-якої точки мережі. Повинен бути якийсь компроміс – так сказати, необхідне й достатнє.

Думаємо, кожний визначає цей мінімум виходячи зі свого життєвого досвіду. Один перестає залишати відкритими всі порти, після того як зламують його встаткування. Іншої починає скрізь включати протоколювання, якщо встаткування раптово вийшло з ладу, а через відсутність журнальних записів зрозуміти причину не вдалося. Тому відзначимо ті особливості налаштування керування, які, на наш погляд, найбільш важливі.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема системи.

Моніторинг мережної інфраструктури

Моніторинг мережних пристроїв містить у собі збір системних повідомлень, контроль доступності, телеметрію мережного пристрою, а також

КПЗ-2025

оповіщення інженера про зміни в мережі. Моніторинг каналів зв'язку організується за протоколом Netflow. Далі розглянемо кожний етап більш докладно.

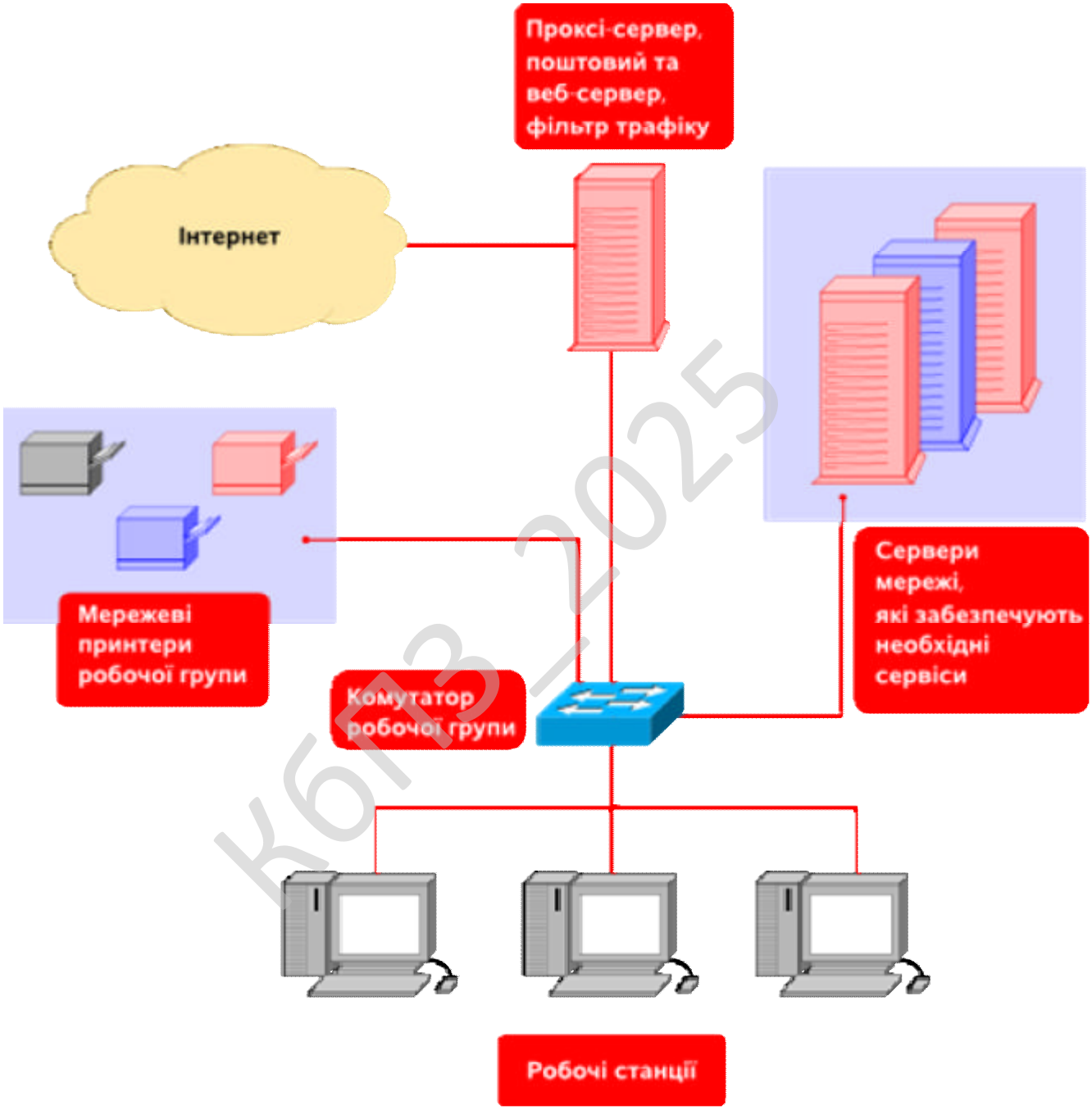

Рисунок 3.1 – Структурна схема системи

Повідомлення syslog і їхній аналіз. Журнали подій фіксують процес роботи мережного пристрою й відбивають його стан. Будь-яке виконане

системою дія відбивається у відповідному системному повідомленні – запису в журналі (балці). Системні повідомлення можуть мати різні рівні деталізації, і залежно від цього в журналі подій виділяються різні рівні протоколювання (логування). Для встаткування Cisco Systems передбачено вісім рівнів: від рівня 0 (Emergencies – повідомлення про непрацездатність системи) до рівня 7 (Debugging – відлагожувальні повідомлення).

Системні повідомлення незамінні при розслідуванні непередбачених мережних проблем. От простий приклад. Мережний адміністратор періодично одержує від користувачів скарги на те, що протягом робочого дня три-чотири рази пропадає зв'язок між центральним офісом і віддаленим. Що робити в такій ситуації?

У першу чергу треба уточнити, коли саме це відбувалося, і подивитися журнали подій мережних пристроїв у часові інтервали, що цікавлять. Припустимо, із записів видно, що на всьому мережному встаткуванні центрального офісу зникає маршрут у віддалений офіс. Аналіз журналів устаткування цього офісу показує, що саме в мнас, що цікавлять, інтервали часу на маршрутизаторі, до якого підключений канал до центрального офісу, з'являлося критичне повідомлення про недостачу оперативної пам'яті на пристрої й потім вироблялося перезавантаження.

Таким чином, по записах у журналі ми змогли локалізувати проблему й зрозуміти причину її появи.

Збір даних за протоколом SNMP. Простий протокол керування мережею (Simple Network Management Protocol, SNMP) є стандартом для обміну керуючою інформацією між мережними пристроями й системою керування мережею (Network Management System, NMS).

Збір даних за протоколом NetFlow. Основні області застосування NetFlow: моніторинг утилізації каналів передачі даних, облік трафіку, проведення аудитів мережної інфраструктури. Ураховувати трафік доводиться насамперед

КПЗ-2025

компаніям – провайдерам зв'язку. При проведенні аудита NetFlow допомагає збирати детальну інформацію про реальний трафіку, переданому по мережі.

Щоб зрозуміти, яким образом NetFlow може допомогти мережному інженерові в керуванні мережею, повернемося до розгляду приклада з періодично, що погіршується якістю, зв'язку (див. підрозділ «Збір даних за протоколом SNMP»). Отже, від користувачів періодично надходять скарги на те, що мережа «гальмує» і телефонія «глючить». За допомогою протоколу SNMP удалося визначити, що завантаження всіх мережних пристроїв на маршруті проходження трафіку перебуває в припустимих межах. Однак на пристрої, до якого підключений канал WAN, надаваний провайдером, завантаження інтерфейсів незвичайно високий для даної ділянки. Лічильники інтерфейсів показують, що кількість переданого/прийнятого трафіку каналу перевищує гарантовану провайдером пропускну здатність.

У даній ситуації протокол NetFlow допоможе зрозуміти, що конкретно відбувається в проблемному каналі. Інженер може побачити, яке додаток надмірно завантажує канал і, головне, між якими парами IP-адрес відбувається передача даних. Цілком імовірно, ці IP-адреси дозволять виявити кінцевих користувачів, що генерують надлишковий трафік. Далі справа залишається за малим. Інженер може уточнити, наскільки необхідно користувачеві настільки неощадливий додаток. Якщо гострої потреби в ньому ні, подібний трафік можна заборонити за допомогою списку доступу на мережному встаткуванні для запобігання повторних проявів проблеми. Якщо ж користувач наполягає на критичності даного сервісу для бізнесу в цілому, мережний інженер повинен підняти питання про розширення пропускної здатності каналу або придбанні нових додаткових каналів. На час вирішення проблеми політики QoS налаштовуються таким чином, щоб трафік спірного додатка не займав весь канал, залишаючи досить ресурсів для передачі як мінімум голосових повідомлень.

Варто відзначити, що збір інформації із протоколу NetFlow створює додаткове навантаження на мережне встаткування, тому не варто знімати дані за

КОПЗ-2025

допомогою NetFlow із всіх вузлів мережної інфраструктури. Рекомендується запитувати тільки дійсно необхідну інформацію про сегменти мережі. До таких ділянок у першу чергу відносяться точки підключення зовнішніх ліній зв'язку: виділених ліній, WAN-каналів і каналів для підключення до Інтернету.

Повідомлення. Всі існуючі комплексні системи моніторингу мережних пристроїв дозволяють налаштовувати повідомлення про події, що відбуваються на пристроях. Повідомлення можна відправляти на підставі переривань SNMP (SNMP trap) або записів syslog у вигляді повідомлень електронної пошти або через шлюз sms. Варто настроїти їхню кореляцію, щоб відправляти саме критичні повідомлення, на які необхідно оперативно реагувати, а також розподілити відповідальність між групами (тип устаткування, рівень помилок, розташування й т.п.) і відправляти інформацію адміністраторам з конкретної групи.

Навіть при відсутності виділеного сервера системи моніторингу з необхідним функціоналом розсилання повідомлень, відправлення повідомлення на електронну пошту при настанні певних подій (syslog, SNMP) можна організувати засобами маршрутизатора. У маршрутизаторах Cisco функціонал Embedded Event Manager (EEM) дозволяє відправити таке повідомлення. Тригером може служити, наприклад, повідомлення syslog про те, що трек ip sla недоступний. Іншими словами, при відмові мережі основного провайдеру й перемиканні на резервний канал, на пошту адміністратора прийде відповідний лист. Ця можливість дуже зручна для невеликих мереж.

Документування мережної інфраструктури

Гарна документація – застава того, що, якщо в мережі щось зламається або буде потрібно внести якісь зміни, всі необхідні дії можна буде провести поза залежністю від зовнішніх умов. Багато компаній зневажають документуванням до того моменту, поки не звільниться співробітник, що обслуговує мережу. При цьому створення документації й підтримка її в актуальному стані, що набагато складніше, найчастіше виконуються вручну, оскільки автоматизувати ці процеси можна лише частково.

КПЗ-2025

Глибина прорoблення документації може залежати як від ступеня складності самої мережі, так і від її призначення. Наприклад, для мереж зі складною топологією повинен бути продуманий план по відновленню працездатності у випадку серйозного збою. Якщо в компанії є чергові адміністратори, для них можуть бути підготовлені інструкції з рішення типових проблем або завдань. До складу документації, на наш погляд, обов'язково повинні входити наступні розділи: доступ до встаткування (імена/паролі, інтерфейси керування, обмеження доступу), фізична й логічна топології мережі, коротке опис роботи основних вузлів. Вся документація повинна зберігатися в захищеному місці, тому що вона містить ключ до всієї мережі.

Однієї з важливих завдань документування є збір актуальної інформації про топологію мережі, що можна зробити як вручну, так і з використанням автоматизованих інструментів. В обох випадках состав засобів, використовуваних для збору інформації, приблизно однаковий.

Де ж взяти дані про поточну топологію? Збір відомостей про фізичну топологію мережі шляхом огляду й інвентаризації безпосередньо на місці ми вже обговорювали. Однак цей очевидний шлях найчастіше складний, трудомісткий і вносить додаткові ризики, адже для перевірки деяких каналів, імовірно, буде потрібно їх відключити. Чи все встаткування буде працювати, як і колись, після такої інвентаризації? Крім того, крім фізичної топології, необхідно внести інформацію про з'єднання й протоколи на канальному й мережному рівнях. Для цього можна використовувати як спеціалізовані протоколи (CDP, LLDP), що надають відомості про сусідні пристрої, так і загальноприйняті службові протоколи й таблиці. Найбільш корисними є таблиці маршрутизації (сусідні маршрутизатори/комутатори, рух трафіку), таблиця MAC-адрес (сусідні пристрої), STP (стан портів), VTP (VLAN), ICMP і ARP (доступність пристрою).

Візуалізувати топологію можна й вручну – в одному зі спеціалізованих пакетів (MS Visio). Однак у цьому випадку оперативно відслідковувати зміни в мережі не вдасться, оскільки будь-яка зміна треба вносити вручну. Особливо

КОПІЯ 2025

складно контролювати зміни при великій кількості встаткування й відсутності твердих регламентів і вимог до внесення змін як у його конфігурацію, так і в документацію. Автоматичне (повне або часткове) складання топології мережі, як правило, входить у функціонала системи моніторингу. Звичайно, при використанні цього методу мережне встаткування прийде підготувати для коректного збору інформації й складання топології, однак він дозволяє зв'язати з топологією всі функції системи моніторингу: відображення стану з'єднань у режимі онлайн, стан пристроїв і їхня доступність, що течуть помилки на пристроях і багато чого іншого. Окрему увагу варто приділити топології бездротової мережі. Для оперативного моніторингу при складанні її топології необхідно вказати розташування точок доступу на плані. Не менш важлива карта покриття мережі. За допомогою системи моніторингу можна подивитися «стандартні» параметри пристрою (клієнти, діапазон, помилки), а при необхідності – відслідковувати положення пристроїв. Можна здійснювати моніторинг не тільки легітимних клієнтів мережі (бездротових клієнтів, RFID-міток і іншого бездротового встаткування), але й пристроїв, що створюють перешкоди в роботі мережі, а також погрози безпеки.

Комплексні системи моніторингу мережної інфраструктури

Список систем моніторингу мережних пристроїв величезний, але дійсно комплексних не так багато. Такі системи мають великий функціонал по моніторингу, налаштуванню мережних пристроїв і аудитові всієї мережі. Вони здатні збирати й аналізувати безліч протоколів (SNMP, syslog, NetFlow, CDP і ін.) і надавати детальну інформацію про роботу мережі. З найпоширеніших можна виділити продукти компаній SolarWinds, ManageEngine, Paessler, Cisco. Варто також відзначити продукти з відкритим кодом, наприклад Nagios і Zabbix.

Nagios надає величезний спектр функцій по моніторингу мережного встаткування й по суті є стандартом де-факто для моніторингу робочої IT-інфраструктури у світі відкритих систем. Недолік цієї системи – налаштування моніторингу в ручному режимі або з попередньою установкою додаткових

КОПЗ-2025

графічних інтерфейсів. Відмінною рисою Zabbix, у порівнянні з Nagios, є наявність зручного графічного інтерфейсу «з коробки».

Як приклад розглянемо систему Cisco Prime Infrastructure (PI). Ця комплексна система моніторингу оснащена безліччю функцій, але їхній опис гідно окремої статті. Тут же приведемо кілька ситуацій, коли система моніторингу придалася на практиці, що, імовірно, тією чи іншою мірою актуально й для інших аналогічних продуктів.

Ситуація 1. Замовник настроїв нового функціонала на маршрутизаторі. Все працює, однак виникли проблеми із сервісами, що використовувалися раніше.

Рішення. Повний архів конфігурації для всіх пристроїв дозволив виявити зміни, внесені в неї за останні три дні. У результаті була знайдена й виправлена помилка в конфігурації.

Ситуація 2. Спостерігається нестабільна робота протоколу DMVPN в одному з нових офісів компанії. Конфігурація типова, розгорнута за допомогою системи шаблонів на PI.

Рішення. При інвентаризації пристрою й порівнянні з іншими виявлено, що була встановлена нестабільна версія ПЗ. Заплановано автоматичне відновлення в неробочий час.

Ситуація 3. Користувач радіотелефону скаржиться на періодичні проблеми зі зв'язком, але на момент звернення все працює добре.

Рішення. Звіт про рівень сигналу протягом дня показав, що дійсно спостерігаються періоди різкого погіршення сигналу. Після виявлення проблемних ділянок був проведений додатковий моніторинг роботи БЛОМ, до повторного прояву проблеми.

Ситуація 4. Співробітник приніс в офіс пристрій 3G і роздає Wi-Fi колегам без обмежень. Щоб не викликати підозр, він використовує корпоративний SSID.

КОПІЯ 2025

Рішення. Система аналізу «сторонніх» пристроїв у бездротовій мережі виявила погрозу по сигнатурі «використання корпоративного SSID». Адміністраторові відправлене повідомлення. По топології мережі з позиціонуванням пристроїв визначене зразкове місце розташування нелегітимної точки доступу. Точка локалізована й виключена.

Ситуація 5. Віддалений співробітник підключився по VPN і скачивает більші обсяги даних з корпоративної мережі, створюючи навантаження на маршрутизатор (шифрування, трафік).

Рішення. Звіт за протоколом NetFlow виявив різке збільшення трафіку, що надходить із одного із серверів корпоративної мережі на хост у віддаленій мережі за протоколом smb/cifs. Налаштовано обмеження пропускної здатності.

Ці прості приклади використання системи моніторингу в мережі показують, як її можливості дозволяють істотно прискорити рішення проблеми й запобігти появі нових. Звичайно, впровадження такої системи, швидше за все, не буде доцільним, якщо в мережі всього 10 пристроїв. Однак при установці в досить великій мережі (у тому числі розподіленої) цей продукт дозволить істотно поліпшити якість її роботи й зменшити витрати на обслуговування.

Хотілося б додати, що одним з найпоширеніших оман є підміна понять: спрощення обслуговування при використанні системи не означає простоти самої системи. Іншими словами, установлюючи систему, замовник найчастіше сподівається, що вже сам факт установки й підключення пристроїв до системи усуне більшість проблем. Як правило, комплексна система – це складний механізм, що вимагає тонкого налаштування й наявності у фахівців навичок по конфігуруванню й розумінню всього циклу роботи встаткування.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Існує не занадто багато способів адміністрування доступу до мережних пристроїв.

КПЗ-2025

Найпростіший з них – збільшення смуги пропущення мережі за рахунок нарощування апаратних можливостей устаткування. Можна використовувати й такі прийоми, як завдання пріоритетів даних, організація черг, запобігання перевантажень і формування трафіку. Керування мережею за заданими правилами в перспективі повинне об'єднати всі ці способи в єдину автоматизовану систему, що буде гарантувати якість послуг абсолютно на всіх ділянках мережі.

Збільшення апаратної потужності, безсумнівно, є найбільш ефективним засобом адміністрування доступу до мережних пристроїв у корпоративній мережі. Тиск із боку конкурентів, необхідність підвищення ефективності виробництва, поява нових технологій, що дозволяють оснащувати спеціалізовані мікросхеми (ASIC) найрізноманітнішими функціями, – все це змушує постачальників комутаційного встаткування для корпоративних мереж викидати на ринок усе більше швидкодіючі пристрої за цінами, порівнянним з вартістю моделей колишнього покоління.

Малоймовірно, що в доступному для огляду майбутньому даний підхід до адміністрування доступу до мережних пристроїв у корпоративних мережах перестане бути пріоритетним. Оскільки в корпоративних мережах вдається забезпечити гарантовану якість послуг, не прибігаючи до дорогої модернізації усього встаткування й серйозних змін у системі керування мережею, мережні адміністратори будуть звертати увагу й на програмні засоби, що дозволяють реалізувати адміністрування доступу до мережних пристроїв.

Отже, найбільше поширення, швидше за все, одержить комбінований підхід. Деякі виробники висловлюються на його користь, затверджуючи, що найкраще збільшувати пропускну здатність мережі не прямо, а за рахунок інтелектуальних можливостей устаткування, що має засоби забезпечення адміністрування доступу до мережних пристроїв. Правда, виробники мережних пристроїв навряд чи можуть бути об'єктивними в цьому питанні, тому що вони зацікавлені в збуті тих самих продуктів, які підтримують гарантовану якість послуг.

КПІ_2025

У глобальних мережах нарощування апаратних потужностей використовується рідше. Звичайно, зниження вартості смуги пропущення зробило б передачу даних по глобальних мережах доступною для більше широкого кола користувачів (і навіть трохи знизило б актуальність впровадження гарантованої якості послуг). Але в найближчому майбутньому вартість смуги пропущення в глобальних мережах буде залишатися досить високою, тому й нарощування апаратної потужності не стане настільки популярним, як у корпоративних мережах.

З рисунку видно, що розроблена система складається з наступних функціональних частин:

– Блок інтерфейсу користувача.

– Блок моніторингу мережі.

– Блок дослідження можливостей механізмів WRED.

– Блок дослідження можливостей механізмів WFQ.

– Блок призначення пріоритетів.

– Блок організації та обслуговування черг.

– Блок управління навантаженням.

– Блок формування трафіку.

– Блок визначення параметрів адміністрування доступу до мережних пристроїв.

– Блок примусового завдання параметрів адміністрування доступу до мережних пристроїв.

– Блок моделювання завантаженості мережі.

Розглянемо ці блоки більш детально.

Блок інтерфейсу користувача

Призначений для реалізації взаємодії користувача, або дослідника з системою.

Блок моніторингу мережі

Призначений для аналізу поточного стану мережі.

КБПВ_2025

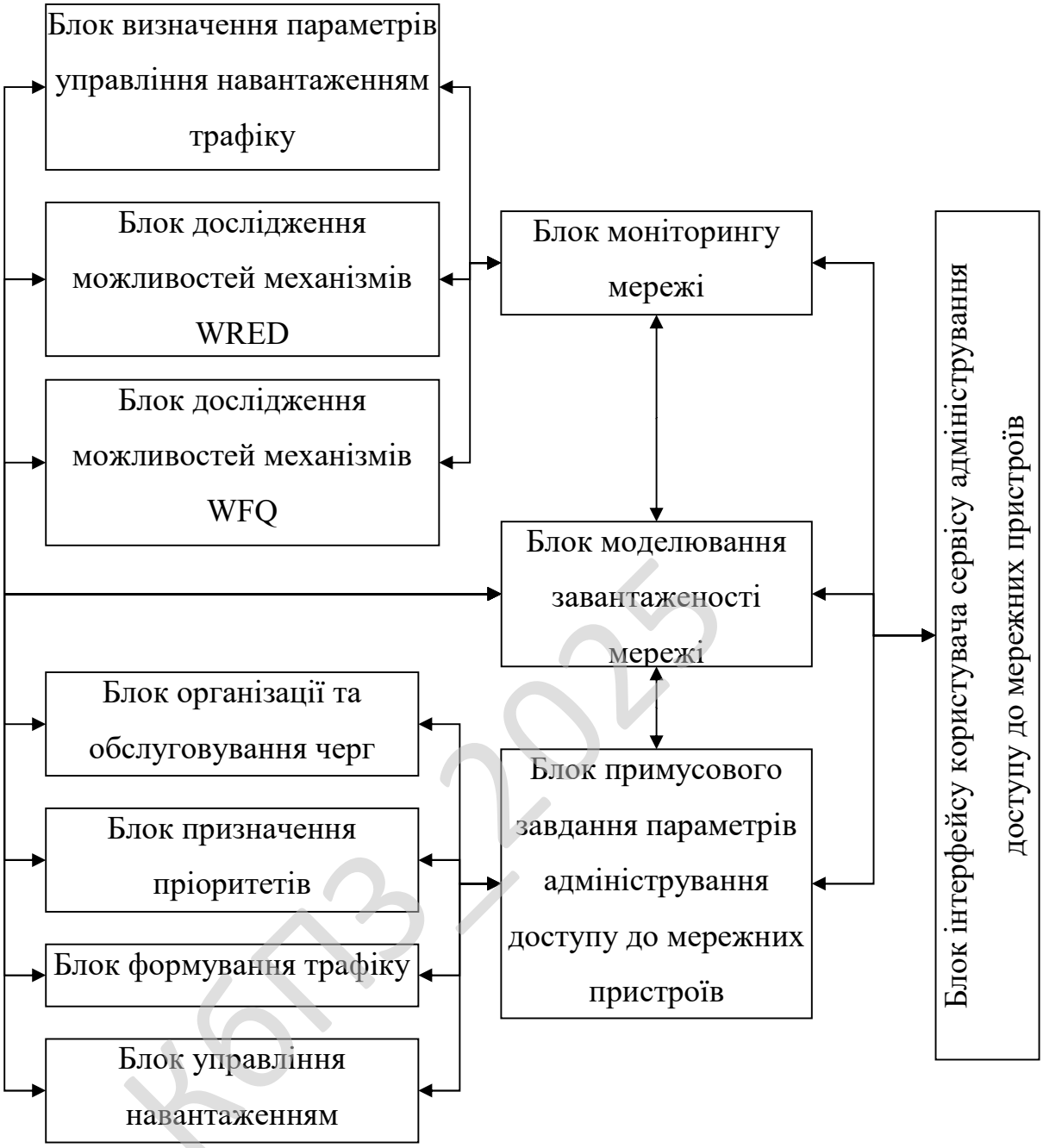

Рисунок 3.2 – Функціональна схема системи

Блок дослідження можливостей механізмів WRED

Одним з методів адміністрування доступу до мережних пристроїв, призначених для забезпечення необхідних вимог до різних потоків даних – запобігання перевантажень (congestion avoidance). Він заснований на обмеженні розмір черги, сигналізуючи джерелам даних про необхідність зменшити

КПЗ 2025

швидкість передачі інформації (WRED – Weighted ranDoSm early detection).

Блок дослідження можливостей механізмів WFQ

Другим з методів адміністрування доступу до мережних пристроїв, призначених для забезпечення необхідних вимог до різних потоків даних – керування перевантаженням (congestion management). Він заснований на присвоєнні квот і пріоритетів потокам, і у випадку перевантаження, потоки одержують якість, обмежену їхньою квотою й пріоритетом (WFQ – Weighted Fair Queuing).

Блок управління навантаженням

Служба адміністрування доступу до мережних пристроїв дає можливість використовувати для керування мережею два важливих механізми – керування в умовах перевантаження й запобігання перевантажень. Перший з них дозволяє кінцевій станції відразу знижувати швидкість передачі даних, коли в мережі починається втрата пакетів. У протоколах TCP/IP і SNA цей механізм підтримується вже протягом декількох років. І хоча сам по собі він не гарантує якості передачі, при його використанні разом з механізмом запобігання перевантажень результати виявляються набагато кращими. У мережах TCP/IP механізм запобігання перевантажень застосовується досить давно, але лише в останні роки він стає стандартом “де-факто” для маршрутизаторів телекомунікаційних мереж і Internet.

Стандартним способом запобігання перевантажень у мережі стало застосування механізму випадкового виділення пакетів (RanDoSm Early Detection, RED). При заповненні черг вище певної критичної оцінки цей механізм змушує маршрутизатор вибирати із черги за випадковим законом деякі пакети й “втрачати” їх. Швидкість передачі даних станціями-відправниками знижується, що й дозволяє уникнути переповнення черги.

Механізм пропорційного випадкового виділення пакетів – WRED (Weighted RED) – можна вважати наступною, більше зробленою “версією” RED. Він передбачає, що вибір пакетів, які повинні “втратитися”, буде відбуватися з обліком їх пріоритезації згідно IP TOS.

КОПІЯ_2025

Блок формування трафіку

Формування трафіку – це загальний термін, яким прийнято позначати різні способи маніпулювання даними для підвищення якості їхньої передачі. Один їхній таких способів – сегментація пакетів. У мережах ATM гарантовано високий рівень адміністрування доступу до мережних пристроїв досягається в тому числі й за рахунок малого розміру переданих пакетів (осередків – у термінології ATM). Максимальний час затримки при передачі будь-якого пакета мережі ATM – це час передачі одного осередку.

Запозичаючи корисні механізми технології ATM, виробники маршрутизаторів і комутаторів починають забезпечувати у своїх продуктах можливість сегментації пакетів. Деякі пристрої, призначені для мереж frame relay, сегментують пакети, передані по каналах глобальних мереж, щоб гарантувати конкретний час передачі й мінімізувати затримки.

Ще один спосіб формування трафіку – його “вирівнювання”. Для таких протоколів, як наприклад, AppleTalk, характерна нерівномірна передача пакетів, що часом приводить до появи в мережі послідовностей або ланцюжків пакетів, а отже – до її перевантаження. Процедура вирівнювання трафіку дозволяє розчленувати ланцюжки шляхом розміщення пакетів у буфері перед їхньою передачею в мережу. Для забезпечення більше рівномірної передачі даних можна також вирівнювати трафік кінцевих вузлів мережі.

Блок визначення параметрів адміністрування доступу до мережних пристроїв

Призначений для визначення існуючих параметрів якості обслуговування (адміністрування доступу до мережних пристроїв). До них відносяться:

– Bandwidth (BW) – смуга пропущення, описує номінальну пропускну здатність середовища передачі інформації, визначає ширину каналу. Виміряється в bit/s (bps), kbit/s (kbps), mbit/s (mbps).

– Delay – затримка при передачі пакета.

– Jitter – коливання (варіація) затримки при передачі пакетів.

– Packet Loss – втрати пакетів. Визначає кількість пакетів, що відкидаються мережею під час передачі.

КПІЗ 2025

Блок примусового завдання параметрів адміністрування доступу до мережних пристроїв

Призначений для примусового завдання одного, або декількох параметрів якості обслуговування (адміністрування доступу до мережних пристроїв). До них відносяться:

– Bandwidth (BW) – смуга пропущення, описує номінальну пропускну здатність середовища передачі інформації, визначає ширину каналу. Виміряється в bit/s (bps), kbit/s (kbps), mbit/s (mbps).

– Delay – затримка при передачі пакета.

– Jitter – коливання (варіація) затримки при передачі пакетів.

– Packet Loss – втрати пакетів. Визначає кількість пакетів, що відкидаються мережею під час передачі.

Блок моделювання завантаженості мережі

Призначений для моделювання завантаженості мережі з визначеним трафіком та заданими параметрами якості обслуговування (адміністрування доступу до мережних пристроїв).

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

Блок призначення пріоритетів

Нарівні з нарощуванням апаратного забезпечення мережі для реалізації адміністрування доступу до мережних пристроїв застосовуються й засобу типу завдання пріоритетів даних і організації черг. Маршрутизатори підтримують ці механізми протягом багатьох років, як і деякі з нових комутаторів для каналів Gigabit Ethernet. Однак ПЗ для керування мережею за заданими правилами, яких необхідно для практичного втілення цієї технології, поки не розроблено. Серед нових комутаторів такого класу можна назвати CoreBuilder 3500, CoreBuilder 9000 і SuperStack II компанії 3Com, пристрою серії Accelar фірми Bay Networks, SmartSwitch Router компанії Cabletron Systems, а також Catalyst 5000 і Catalyst 8000 виробництва Cisco.

Способи пріоритезації даних можна умовно підрозділити на явні й неявні.

КБПЗ-2025

При неявному призначенні пріоритетів маршрутизатор або комутатор автоматично привласнює послугам відповідні рівні, виходячи із заданих адміністратором мережі критеріїв (наприклад, типу додатка для застосовуваного протоколу передачі або адреси джерела). Кожний вхідний пакет аналізується (фільтрується) на відповідність цим критеріям. Механізм неявної пріоритезації підтримують практично всі маршрутизатори.

Деякі комутатори теж здатні задавати пріоритети, але мають обмежений набір функцій. Так, комутатори можуть забезпечувати пріоритезацію даних по типу віртуальної корпоративної мережі, адресі джерела або адресата, але не використовують інформацію більш високого рівня (протокол передачі або тип додатка). Розроблювальні в цей час системи керування мережею за заданими правилами дозволять реалізувати більше зроблені схеми пріоритезації даних при роботі з такими комутаторами.

При явній пріоритезації даних користувач або додаток запитує певний рівень служби, а комутатор або маршрутизатор намагається задовольнити запит. Імовірно, самим популярним механізмом явної пріоритезації стане протокол IP Precedence (протокол старшинства), що одержав другу назву IP TOS (IP Type Of Service), – один з розділів четвертої версії протоколу IP.

IP TOS резервує спеціальне поле в заголовку пакета, де можуть бути зазначені ознаки адміністрування доступу до мережних пристроїв, що визначають час затримки, швидкість пропущення й рівень надійності передачі пакета. Однак найдеться небагато популярних додатків – за винятком мультимедійного ПЗ, – у які реалізована підтримка протоколу IP TOS.

Зараз розробляється протокол резервування ресурсів RSVP, що передбачає більше складний, чим в IP TOS, механізм передачі від додатка до машрутизатору запиту на гарантовану якість послуг. Як і IP TOS, протокол RSVP поки не одержав широкої підтримки розроблювачів – він реалізований лише в окремих типах маршрутизаторів. Поширення RSVP стримується через те, що не вирішені деякі питання, пов'язані із сумісністю різних мереж. До того ж застосування

КОПЗ_2025

RSVP значно збільшує навантаження на маршрутизатори й може привести до зниження швидкодії цих пристроїв.

Видимо, у доступному для огляду майбутньому неявна пріоритезація, не потребуюча серйозних обчислювальних потужностей маршрутизатора, залишиться більше популярною, чим явна. Крім того, при явному завданні пріоритетів значно ускладнюється керування мережею. Кінцеві користувачі, швидше за все, будуть набудовувати своє програмне забезпечення на запит найвищого з можливих рівнів послуг. Відповідно, адміністраторові мережі прийдеться розробляти правила керування користувачами й, можливо, навіть побудовувати служби з гарантованою якістю для кожного користувача окремо.

Блок організації та обслуговування черг

Після того як переданим по мережі даним призначені відповідні пріоритети (за допомогою явних або неявних методів), потрібно визначити порядок передачі цих даних, задавши алгоритм обслуговування черг із необхідною якістю (рівнем адміністрування доступу до мережних пристроїв). По суті, черги являють собою області пам'яті комутатора або маршрутизатора, у яких групуються пакети з однаковими пріоритетами передачі. Алгоритм обслуговування черги визначає порядок, у якому відбувається передача пакетів, що зберігаються в ній. Зміст застосування всіх алгоритмів зводиться до того, щоб забезпечити найкраще обслуговування трафіку з більш високим пріоритетом за умови, що й пакету з низьким пріоритетом гарантується відповідна увага.

При використанні способів завдання явних і неявних пріоритетів алгоритм обробки черг визначає порядок їхнього обслуговування. Відповідно до цього алгоритму на кожні два пакети, переданих у мережу із черги 1 (з високим пріоритетом) доводиться по одному пакету із черг 2 і 3. Пакети з однаковими пріоритетами передаються за принципом FIFO (”першим прийшов – першим вийшов”).

Якщо в мережі виникає перевантаження, служба черг не гарантує своєчасного досягнення пункту призначення найбільш важливими даними.

КБПЗ_2025

Гарантується лише те, що ці пакети будуть передані раніше, ніж ті, що мають більш низький пріоритет.

Сучасні служби адміністрування доступу до мережних пристроїв вирішують таке завдання за рахунок резервування смуги пропущення. Кожній із черг (або їхніх груп) виділяється заздалегідь задана величина смуги пропущення, що гарантує певну смугу пропущення для черги з більш високим пріоритетом. Для критичних ситуацій, коли обсяг даних у черзі перевищує розміри смуги пропущення, в алгоритмах обслуговування звичайно передбачається передача трафіку з високим пріоритетом на смугу пропущення, “яка приналежить” чергам з низьким пріоритетом, і навпаки. Найпростіші алгоритми обслуговують кожну чергу за принципом FIFO. При цьому передача кадрів великого розміру, що мають високий пріоритет, може приводити до затримок трафіку іншого додатка з настільки ж високим пріоритетом, але меншим обсягом.

У більше складних алгоритмах уживає спроба “справедливої” обробки черг. Наприклад, алгоритм рівномірного пропорційного (або зваженого) обслуговування (WFQ – Weighted Fair Queuing), розроблений компанією Cisco, підрозділяє додатки на потребуючі великої й малої ширини смуги пропущення, а сама смуга пропущення розподіляється між всіма додатками нарівно. Слід зазначити, що основні виробники маршрутизаторів самі розробляють алгоритми обслуговування черг і використовують для їхнього опису власну термінологію.

Істотним недоліком сучасних маршрутизаторів і комутаторів є те, що вони підтримують мале число черг. Найчастіше виробники організують служби адміністрування доступу до мережних пристроїв, що використовують чотири черги, хоча чим більше черг, тим більше різних пріоритетів можна привласнити переданим пакетам і тим “більш справедливо” розподілити смугу пропущення між додатками. Наприклад, адміністратор у стані задати пріоритети таким чином, щоб перевага при передачі віддавалося пакетам, адресованим на більше віддалені вузли.

КПЗ-2025

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврської дипломної роботи, наведена на рисунку 3.3.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

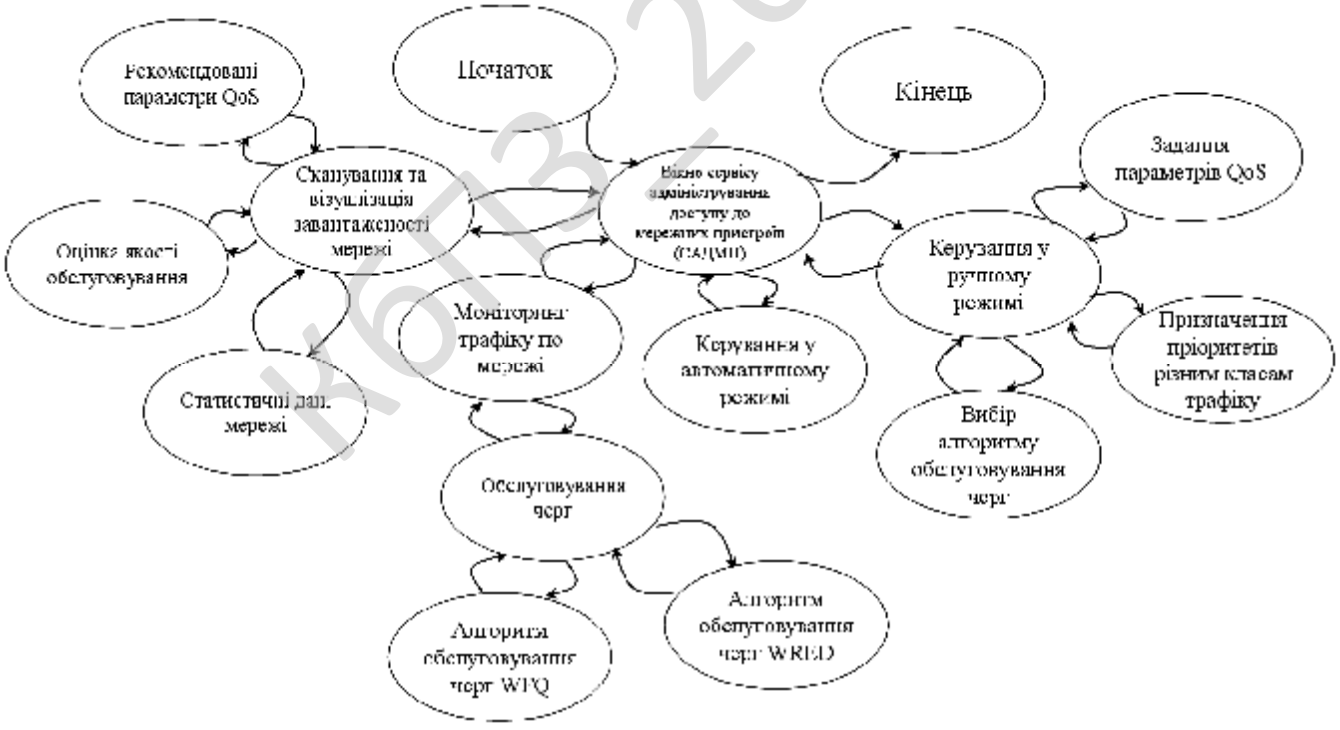

Рисунок 3.3 – Діаграма взаємодії процесів

КПІ ІВ 2025

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

КбПЗ_2025

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю сервісу адміністрування доступу до мережних пристроїв.

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

КПЗ_2025

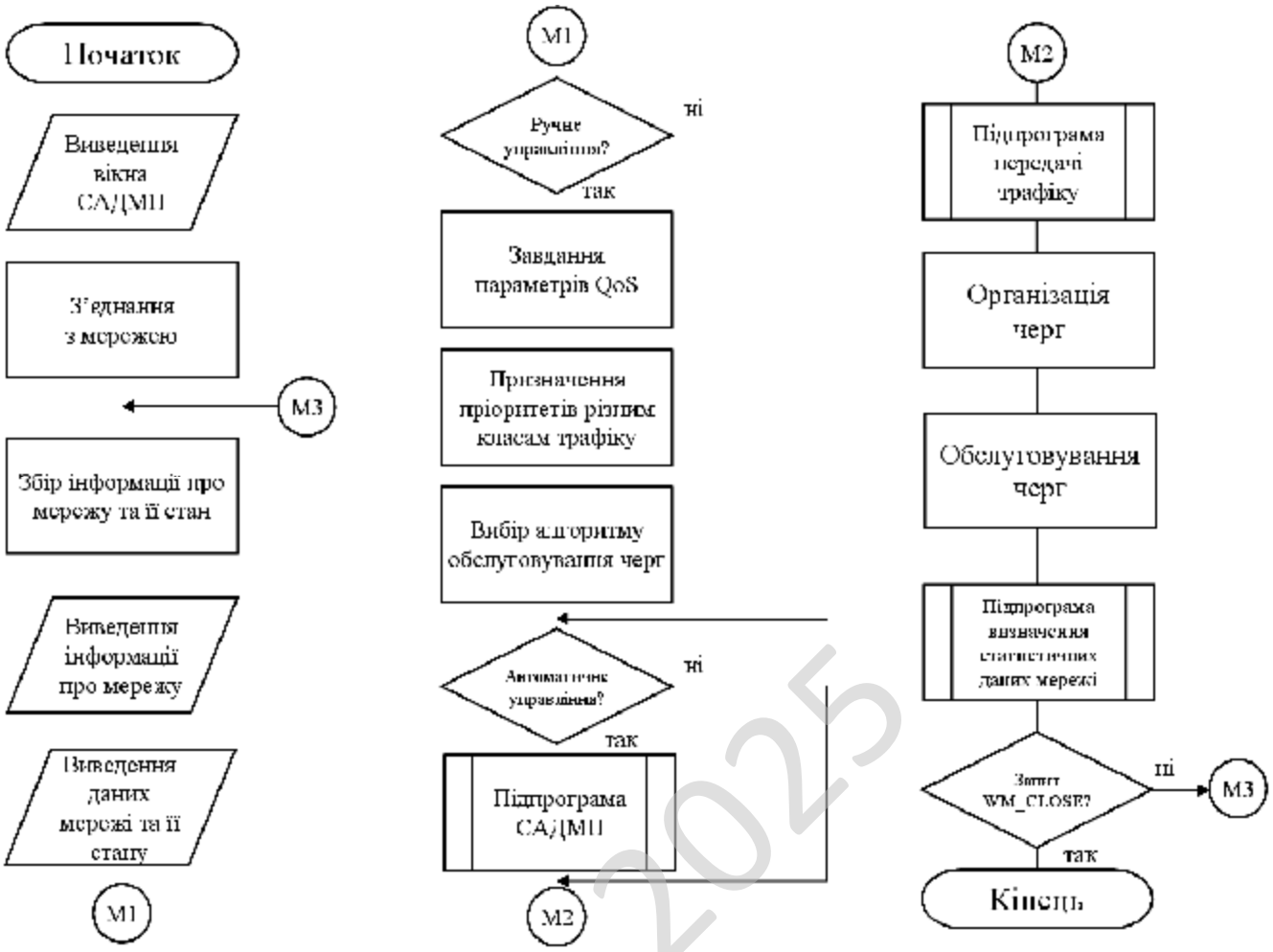

Рисунок 4.1 – Блок-схема основної програми

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

КБПЗ_2025

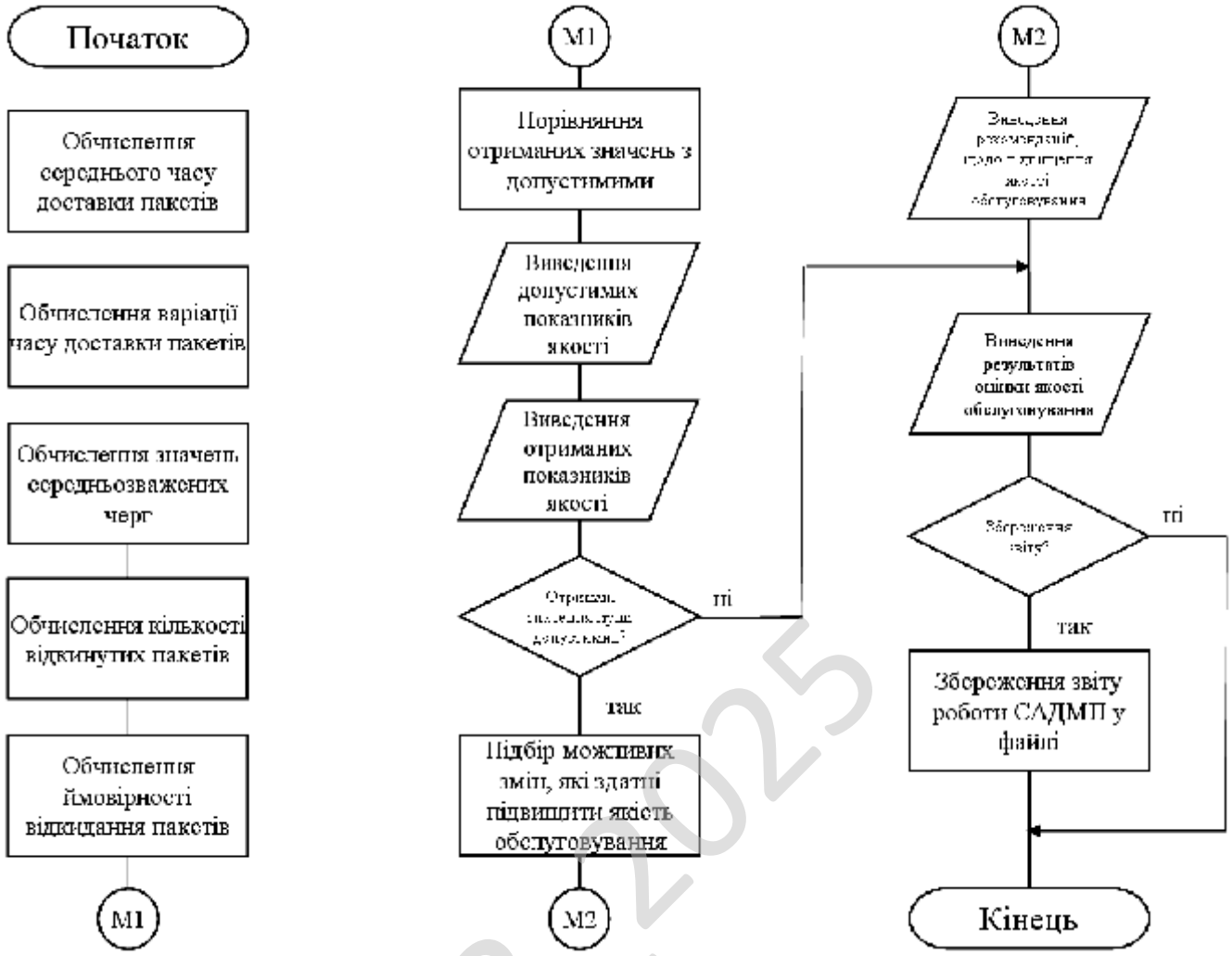

Рисунок 4.2 – Блок-схема роботи підпрограми

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка

КПЗ_2025

потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

КПІ – 2025

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

КПЗ_2025

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

Опис системи

Система адміністрування доступу до мережних пристроїв реалізується за допомогою мови програмування Python. Вона забезпечує управління підключеннями, контроль доступу та моніторинг активності користувачів у мережі. Архітектура включає серверну частину, клієнтські модулі та базу даних для зберігання інформації про користувачів і їхні права доступу.

Серверна частина відповідає за автентифікацію та авторизацію користувачів. Вона використовує бібліотеку Flask для реалізації веб-інтерфейсу та API-запитів. Основний механізм перевірки прав доступу ґрунтується на токенах, які генеруються після успішного входу та зберігаються у базі даних PostgreSQL.

КБПЗ_2025

Клієнтська частина представлена модулями, що працюють на пристроях адміністраторів та користувачів. Вона реалізує взаємодію з сервером через HTTPS-запити та використовує бібліотеку Requests для обміну даними. Додатково впроваджується підтримка роботи через SSH, що дозволяє виконувати команди на мережних пристроях.

База даних PostgreSQL містить таблиці користувачів, їхні рівні доступу, логування підключень та зміни в мережевій конфігурації. Оптимізація бази даних включає використання індексів для швидкого пошуку користувачів та їхніх прав.

Основні модулі системи:

– Модуль автентифікації перевіряє введені користувачем дані, генерує токен та зберігає його в базі даних. Для шифрування паролів використовується бібліотека bcrypt.

– Модуль авторизації контролює права доступу на основі ролей користувачів. Для кожного користувача встановлюються правила, які визначають його можливості у системі.

– Модуль моніторингу веде журнал підключень, зберігає інформацію про зміни у конфігурації мережі та аналізує підозрілу активність.

– Модуль адміністрування дозволяє керувати обліковими записами, додавати нових користувачів та змінювати їхні права.

– Модуль підключення до пристроїв реалізує доступ до мережних пристроїв через протоколи SSH або Telnet. Використовується бібліотека Paramiko для роботи з SSH-з’єднаннями.

– Модуль API забезпечує взаємодію між серверною частиною та клієнтськими додатками, використовуючи REST-архітектуру.

Розрахунки продуктивності системи

Припустимо, що система обслуговує 1000 користувачів, з яких одночасно працюють 200. Запити автентифікації займають 0.2 секунди, а перевірка прав доступу – 0.05 секунди.

Навантаження на сервер при такій кількості користувачів залишається в

КПЗ-2025

межах допустимих значень. Швидкість обробки запитів дозволяє забезпечити відгук сервера не більше 0.5 секунди навіть у пікові години.

Система підтримує логування дій користувачів, що дозволяє відслідковувати всі зміни у мережі. Дані зберігаються у базі PostgreSQL, а для швидкого пошуку реалізовано індексацію. Оптимізація запитів до бази даних забезпечує мінімальний час відповіді, що важливо при обробці великої кількості запитів.

Безпека системи забезпечується шифруванням даних, використанням захищених протоколів та двофакторною аутентифікацією. Паролі зберігаються у хешованому вигляді, а доступ до системи здійснюється тільки через захищене з'єднання.

Впровадження даної системи у навчальному закладі забезпечує контроль за підключенням до мережі, підвищує рівень безпеки та спрощує адміністрування. Автоматизація процесів зменшує ризик несанкціонованого доступу та покращує ефективність роботи IT-відділу.

Частина вихідного коду системи

```
import bcrypt
import jwt
import datetime
import psycopg2
import paramiko
from flask import Flask, request, jsonify
from functools import wraps

# Ініціалізація Flask-додатку
app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key'

# Підключення до бази даних
conn = psycopg2.connect(
    dbname="network_access",
    user="admin",
    password="password",
    host="localhost")
cursor = conn.cursor()
```

КПЗ_2025

```
        # Функція хешування паролів
        def hash_password(password):
        return                                  bcrypt.hashpw(password.encode('utf-8'),
bcrypt.gensalt()).decode('utf-8')

        # Функція перевірки пароля
        def check_password(password, hashed):
        return bcrypt.checkpw(password.encode('utf-8'), hashed.encode('utf-8'))

        # Функція генерації токена
        def generate_token(user_id):
        payload = {
            'user_id': user_id,
            'exp': datetime.datetime.utcnow() + datetime.timedelta(hours=1)}
        return jwt.encode(payload, app.config['SECRET_KEY'], algorithm='HS256')

        # Декоратор для перевірки токена
        def token_required(f):
        @wraps(f)
        def decorated(*args, **kwargs):
            token = request.headers.get('Authorization')
            if not token:
                return jsonify({'message': 'Токен відсутній'}), 401
            try:
        data = jwt.decode(token, app.config['SECRET_KEY'], algorithms=['HS256'])
        cursor.execute("SELECT id FROM users WHERE id = %s", (data['user_id'],))
                if cursor.fetchone() is None:
                    return jsonify({'message': 'Невірний токен'}), 401
            except jwt.ExpiredSignatureError:
                return jsonify({'message': 'Термін дії токена минув'}), 401
            except jwt.InvalidTokenError:
                return jsonify({'message': 'Невірний токен'}), 401
            return f(*args, **kwargs)
        return decorated

        # Реєстрація нового користувача
        @app.route('/register', methods=['POST'])
        def register():
        data = request.json
        hashed_password = hash_password(data['password'])
        cursor.execute(
```

КОПЗ – 2025

```
            "INSERT INTO users (username, password) VALUES (%s, %s)",
            (data['username'], hashed_password)
        )
        conn.commit()
        return jsonify({'message': 'Користувач зареєстрований'})

        # Авторизація користувача
        @app.route('/login', methods=['POST'])
        def login():
        data = request.json
        cursor.execute("SELECT id, password FROM users WHERE username = %s",
(data['username'],))
        user = cursor.fetchone()
        if user and check_password(data['password'], user[1]):
            token = generate_token(user[0])
            return jsonify({'token': token})
        return jsonify({'message': 'Невірний логін або пароль'}), 401

        # Отримання списку користувачів (адміністраторський доступ)
        @app.route('/users', methods=['GET'])
        @token_required
        def get_users():
        cursor.execute("SELECT id, username FROM users")
        users = cursor.fetchall()
        return jsonify({'users': [{'id': u[0], 'username': u[1]} for u in users]})

        # Підключення до мережного пристрою через SSH
        def ssh_connect(host, username, password, command):
        client = paramiko.SSHClient()
        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        try:
            client.connect(host, username=username, password=password)
            stdin, stdout, stderr = client.exec_command(command)
            output = stdout.read().decode()
            client.close()
            return output
        except Exception as e:
            return str(e)

        # Виконання команди на мережному пристрої
        @app.route('/execute', methods=['POST'])
        @token_required
```

КОПЗ_2025

```
        def execute_command():
        data = request.json
        result = ssh_connect(data['host'], data['username'], data['password'],
data['command'])
        return jsonify({'output': result})

        # Запуск сервера
        if __name__ == '__main__':
        app.run(debug=True)
```

Основні особливості коду:

1. Безпека – використовується хешування паролів (bcrypt) та токени для автентифікації (JWT).

2.База даних – PostgreSQL з таблицею користувачів.

3. Авторизація – перевірка прав доступу через токени.

4. Підключення до пристроїв – реалізовано через SSH з використанням бібліотеки Paramiko.

5. API – сервер працює через Flask і дозволяє адмініструвати користувачів та виконувати команди на мережних пристроях.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Lucifer.

Алгоритм Lucifer являє собою мережу перестановок і підстановок, його основні блоки нагадують блоки алгоритму DES. В DES результат функції f складається операцією XOR із входом попереднього раунду, утворюючи вхід наступного раунду.

В S-блоках алгоритму Lucifer 4-бітові входи й виходи, вхід S-блоків являє собою перетасований вихід S-блоків попереднього раунду, входом S-блоків першого раунду служить відкритий текст.

КОПЗ_2025

Для вибору використовуваного S-блоку із двох можливих використовується біт ключа. (Lucifer реалізує все це в єдиному T-блоці з 9 бітами на вході й 8 бітами на виході).

На відміну від алгоритму DES, половини блоку між раундами не переставляються, та й саме поняття половини блоку в алгоритмі Lucifer не використовується.

У цього алгоритму 16 раундів, 128-бітові блоки й більше проста, чим в DES, схема розгорнення ключа.

Блок тексту розглядається як ненегативне ціле число, або як кілька незалежних ненегативних цілих чисел.

Довжина блоку завжди вибирається рівною ступеню двійки. У алгоритмі Lucifer використовуються наступні типи операцій:

– Таблична підстановка, при якій група біт відображається в іншу групу біт. Це так звані S-box.

– Переміщення, за допомогою якого біти повідомлення переупорядковуються.

– Операція додавання по модулю 2, позначувана XOR або $\oplus$.

– Операція додавання по модулю 2^{32} або по модулю 2^{16}.

– Циклічне зрушення на деяке число біт.

Ці операції циклічно повторюються в алгоритмі, створюючи так звані раунди.

Входом кожного раунду є вихід попереднього раунду й ключ, що отриманий по певному алгоритму із ключа шифрування K.

Ключ раунду називається підключем. Алгоритм шифрування може бути представлений у такий спосіб:

КОПЗ_2025

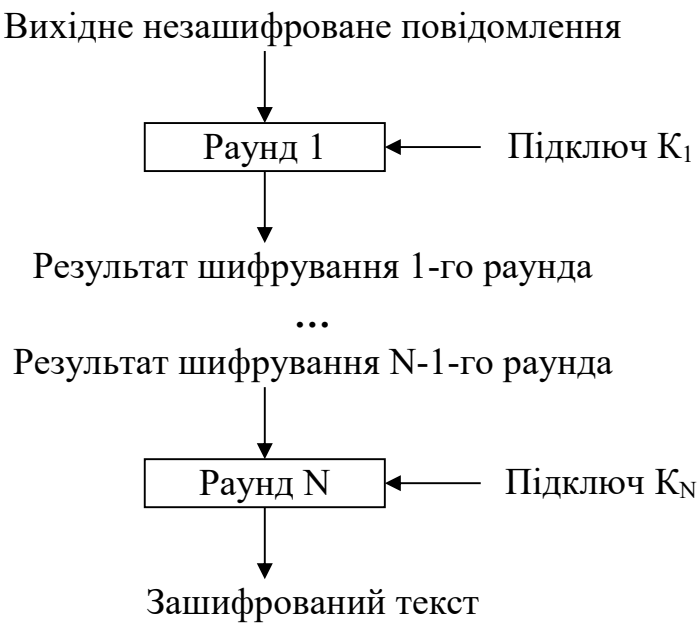

Рисунок 4.3 – Структура алгоритму алгоритмі Lucifer

КбПЗ_2025

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

Розроблене програмне забезпечення сервісу адміністрування доступу до мережних пристроїв складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Управління навантаженням; Моніторинг; Довідка.

– Інформаційна панель відображення показників.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

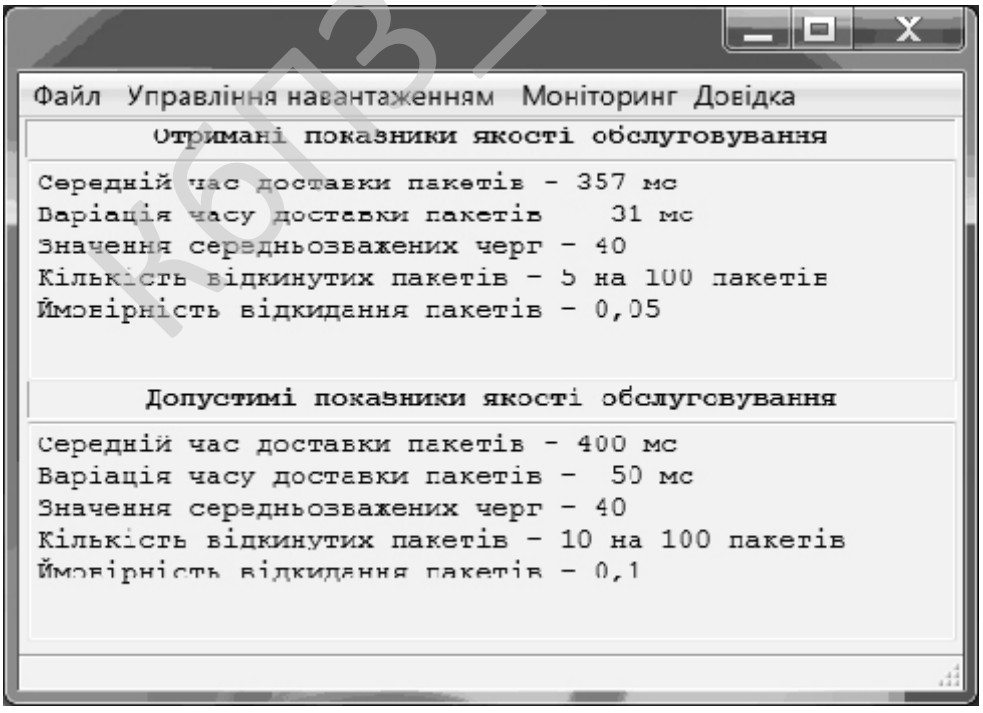

Рисунок 5.1 – Головне вікно розробленого ПЗ

КПЗ – 2025

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

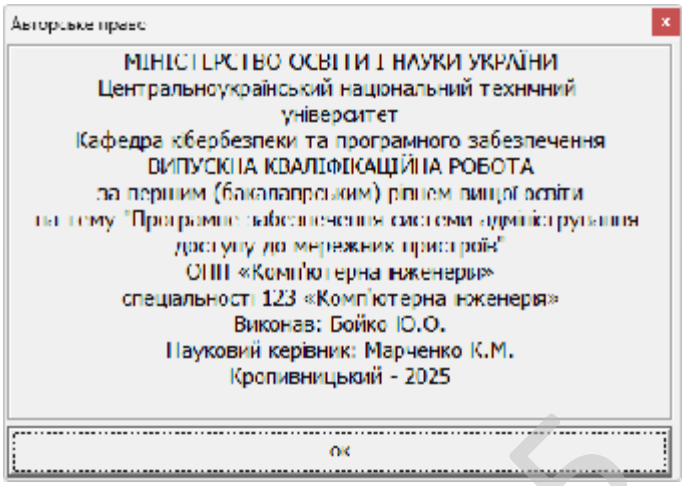

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

КПЗ_2025

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом чорної скриньки**.** Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють: Як виконуються функції програми; Як приймаються вихідні дані; Як виробляються результати; Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе. Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми. Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT. Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

КПЗ-2025

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості. Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

КБПЗ-2025

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи адміністрування доступу до мережних пристроїв.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем адміністрування доступу до мережних пристроїв.

– Досліджена система адміністрування доступу до мережних пристроїв.

– На основі отриманих результатів досліджень створена програмна реалізація системи адміністрування доступу до мережних пристроїв.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання адміністрування доступу до мережних пристроїв.

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

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

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи адміністрування доступу до мережних пристроїв. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної

КПЗ-2025

техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Lucifer.

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

КбПЗ_2025

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

2. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

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

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

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

6. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

7. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

8. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

КБПЗ-2025

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

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

11. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

12. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

13. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

14. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

15. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

16. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties».

КОПЗ-2025

International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

17. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

18. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

19. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

20. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

21. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

22. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

23. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising

КОПЗ – 2025

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

24. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

25. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

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

27. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

28. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

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

30. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного

КБПЗ-2025

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

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

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

33. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

34. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

35. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

36. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

КОПЗ-2025

37. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

38. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

39. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

40. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

41. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

42. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

43. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

КПІ – 2025

44. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

45. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

46. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

47. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

48. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

49. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. - 2016. - С. 121-127.

50. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. - Випуск 2 (46) - Х.: ХУПС - 2016. - С. 146-149.

51. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

КБПЗ – 2025

Додаток А
(обов’язковий)

Технічне завдання

Зміст

КОПЗ–2025

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи адміністрування доступу до мережних пристроїв.

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. № 47-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи адміністрування доступу до мережних пристроїв.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

КБПЗ_2025

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи адміністрування доступу до мережних пристроїв;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

КПЗ_2025

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

КбПЗ – 2025

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 у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 77 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).

КБПЗ_2025

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.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 3.06.2025 р.

КбПЗ_2025

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Марченко К.М.

Програмне забезпечення системи адміністрування доступу до мережних пристроїв

КбПЗ_2025

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 24

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
#Імпорт бібліотек, необхідних для роботи системи адміністрування
import sqlite3
import datetime
import time
import random
import sys
import os

#Клас для роботи з базою даних адміністрування доступу
class AccessControlDB:
    def __init__(self, db_file):
        #Ініціалізація з'єднання з базою даних
        self.db_file = db_file
        self.connection = sqlite3.connect(self.db_file)
        self.cursor = self.connection.cursor()
        self.initialize_tables()

    def initialize_tables(self):
        #Створення таблиці користувачів
        self.cursor.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER
PRIMARY KEY AUTOINCREMENT, username TEXT UNIQUE, password TEXT, role TEXT)")
        self.connection.commit()
        #Створення таблиці мережних пристроїв
        self.cursor.execute("CREATE TABLE IF NOT EXISTS devices (id INTEGER
PRIMARY KEY AUTOINCREMENT, device_name TEXT, ip_address TEXT, device_type
TEXT)")
        self.connection.commit()
        #Створення таблиці логів доступу
        self.cursor.execute("CREATE TABLE IF NOT EXISTS access_logs (id INTEGER
PRIMARY KEY AUTOINCREMENT, user_id INTEGER, device_id INTEGER, access_time TEXT,
action TEXT)")
        self.connection.commit()
        #Створення таблиці конфігурацій пристроїв
        self.cursor.execute("CREATE TABLE IF NOT EXISTS device_configs (id
INTEGER PRIMARY KEY AUTOINCREMENT, device_id INTEGER, config_text TEXT)")
        self.connection.commit()
        #Створення таблиці прав доступу користувачів до пристроїв
        self.cursor.execute("CREATE TABLE IF NOT EXISTS user_permissions (id
INTEGER PRIMARY KEY AUTOINCREMENT, user_id INTEGER, device_id INTEGER,
permission TEXT)")
        self.connection.commit()

    def add_user(self, username, password, role):
        #Додавання нового користувача до таблиці користувачів
        try:
            self.cursor.execute("INSERT INTO users (username, password, role)
VALUES (?, ?, ?)", (username, password, role))
            self.connection.commit()
            return True
        except sqlite3.IntegrityError:
            return False

    def add_device(self, device_name, ip_address, device_type):
        #Додавання нового мережного пристрою до таблиці пристроїв
        try:
            self.cursor.execute("INSERT INTO devices (device_name, ip_address,
device_type) VALUES (?, ?, ?)", (device_name, ip_address, device_type))
            self.connection.commit()
            return True
        except sqlite3.IntegrityError:
            return False

    def log_access(self, user_id, device_id, action):
        #Запис події доступу до таблиці логів
        access_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

КОПІЯ 2025

```
        self.cursor.execute("INSERT INTO access_logs (user_id, device_id,
access_time, action) VALUES (?, ?, ?, ?)", (user_id, device_id, access_time,
action))
        self.connection.commit()

    def add_device_config(self, device_id, config_text):
        #Додавання конфігурації для мережного пристрою
        self.cursor.execute("INSERT INTO device_configs (device_id, config_text)
VALUES (?, ?)", (device_id, config_text))
        self.connection.commit()

    def add_user_permission(self, user_id, device_id, permission):
        #Надання прав доступу користувачу для конкретного пристрою
        self.cursor.execute("INSERT INTO user_permissions (user_id, device_id,
permission) VALUES (?, ?, ?)", (user_id, device_id, permission))
        self.connection.commit()

    def get_users(self):
        #Отримання списку всіх користувачів з бази даних
        self.cursor.execute("SELECT * FROM users")
        return self.cursor.fetchall()

    def get_devices(self):
        #Отримання списку всіх мережних пристроїв з бази даних
        self.cursor.execute("SELECT * FROM devices")
        return self.cursor.fetchall()

    def get_access_logs(self):
        #Отримання всіх логів доступу з бази даних
        self.cursor.execute("SELECT * FROM access_logs")
        return self.cursor.fetchall()

    def get_device_configs(self):
        #Отримання всіх конфігурацій пристроїв з бази даних
        self.cursor.execute("SELECT * FROM device_configs")
        return self.cursor.fetchall()

    def get_user_permissions(self):
        #Отримання всіх прав доступу користувачів до пристроїв
        self.cursor.execute("SELECT * FROM user_permissions")
        return self.cursor.fetchall()

    def verify_user(self, username, password):
        #Перевірка облікових даних користувача
        self.cursor.execute("SELECT * FROM users WHERE username=? AND
password=?", (username, password))
        return self.cursor.fetchone()

    def get_user_by_id(self, user_id):
        #Отримання даних користувача за його ідентифікатором
        self.cursor.execute("SELECT * FROM users WHERE id=?", (user_id,))
        return self.cursor.fetchone()

    def get_device_by_id(self, device_id):
        #Отримання даних пристрою за його ідентифікатором
        self.cursor.execute("SELECT * FROM devices WHERE id=?", (device_id,))
        return self.cursor.fetchone()

    def close(self):
        #Закриття з'єднання з базою даних
        self.connection.close()

#Клас для адміністрування доступу до мережних пристроїв
class NetworkAccessAdmin:
    def __init__(self, db_file="network_access.db"):
        #Ініціалізація системи адміністрування з вказаною базою даних
        self.db = AccessControlDB(db_file)
        self.current_user = None
```

КОПЗ_2025

```
    def register_user(self):
        #Реєстрація нового користувача в системі
        print("Реєстрація нового користувача")
        username = input("Введіть нове ім'я користувача: ")
        password = input("Введіть новий пароль: ")
        role = input("Введіть роль (admin/user): ")
        if self.db.add_user(username, password, role):
            print("Користувача успішно додано.")
        else:
            print("Користувач вже існує або сталася помилка.")

    def add_network_device(self):
        #Додавання нового мережного пристрою до системи
        print("Додавання мережного пристрою")
        device_name = input("Введіть назву пристрою: ")
        ip_address = input("Введіть IP-адресу: ")
        device_type = input("Введіть тип пристрою: ")
        if self.db.add_device(device_name, ip_address, device_type):
            print("Пристрій успішно додано.")
        else:
            print("Не вдалося додати пристрій або він вже існує.")

    def add_device_configuration(self):
        #Додавання конфігурації для мережного пристрою
        print("Додавання конфігурації пристрою")
        try:
            device_id = int(input("Введіть ID пристрою: "))
        except ValueError:
            print("Невірний ID пристрою.")
            return
        config_text = input("Введіть текст конфігурації: ")
        self.db.add_device_config(device_id, config_text)
        print("Конфігурацію пристрою додано.")

    def assign_user_permission(self):
        #Надання прав доступу користувачу для пристрою
        print("Надання прав доступу")
        try:
            user_id = int(input("Введіть ID користувача: "))
            device_id = int(input("Введіть ID пристрою: "))
        except ValueError:
            print("ID має бути числовим.")
            return
        permission = input("Введіть право доступу (read/write/admin): ")
        self.db.add_user_permission(user_id, device_id, permission)
        print("Права доступу успішно надано.")

    def login(self):
        #Вхід користувача в систему
        print("Вхід користувача")
        username = input("Введіть ім'я користувача: ")
        password = input("Введіть пароль: ")
        user = self.db.verify_user(username, password)
        if user:
            self.current_user = user
            print("Вхід успішний. Ласкаво просимо,", username)
        else:
            print("Невірні облікові дані. Спробуйте ще раз.")

    def list_devices(self):
        #Виведення списку мережних пристроїв
        print("Список мережних пристроїв:")
        devices = self.db.get_devices()
        for device in devices:
            print("ID:", device[0], "Назва:", device[1], "IP:", device[2],
"Тип:", device[3])

    def list_users(self):
        #Виведення списку користувачів
```

КОПЗ – 2025

```
        print("Список користувачів:")
        users = self.db.get_users()
        for user in users:
            print("ID:", user[0], "Ім'я:", user[1], "Роль:", user[3])

    def list_access_logs(self):
        #Виведення списку логів доступу
        print("Логи доступу:")
        logs = self.db.get_access_logs()
        for log in logs:
            print("ID логу:", log[0], "ID користувача:", log[1], "ID пристрою:",
log[2], "Час:", log[3], "Дія:", log[4])

    def simulate_device_access(self):
        #Симуляція доступу до мережного пристрою
        if not self.current_user:
            print("Будь ласка, увійдіть в систему для доступу до пристроїв.")
            return
        self.list_devices()
        try:
            device_id = int(input("Введіть ID пристрою для доступу: "))
        except ValueError:
            print("Невірний ввід. ID пристрою має бути числовим.")
            return
        device = self.db.get_device_by_id(device_id)
        if device:
            print("Отримання доступу до пристрою:", device[1])
            time.sleep(1)
            status = random.choice(["успішно", "з невдачею"])
            print("Доступ до пристрою", status)
            self.db.log_access(self.current_user[0], device_id, "Спроба доступу
" + status)
        else:
            print("Пристрій не знайдено.")

    def generate_access_report(self):
        #Генерація детального звіту доступу
        print("Генерація звіту доступу...")
        logs = self.db.get_access_logs()
        report = {}
        for log in logs:
            user_id = log[1]
            if user_id not in report:
                report[user_id] = 0
            report[user_id] += 1
        for user_id, count in report.items():
            user = self.db.get_user_by_id(user_id)
            if user:
                print("Користувач:", user[1], "має", count, "записів логів
доступу.")
            else:
                print("Невідомий користувач з ID:", user_id, "має", count,
"записів.")

    def network_device_scan(self):
        #Симуляція сканування мережі для визначення статусу пристроїв
        print("Сканування мережних пристроїв...")
        devices = self.db.get_devices()
        for device in devices:
            status = random.choice(["онлайн", "офлайн", "невідомо"])
            print("Пристрій:", device[1], "Статус:", status)
            time.sleep(0.5)

    def admin_menu(self):
        #Меню для адміністраторів з повним набором функцій
        while True:
            print("\nМеню адміністратора:")
            print("1. Реєстрація нового користувача")
            print("2. Додавання мережного пристрою")
```

КБПВ_2025

```
            print("3. Додавання конфігурації пристрою")
            print("4. Надання прав доступу користувачу")
            print("5. Виведення списку користувачів")
            print("6. Виведення списку пристроїв")
            print("7. Симуляція доступу до пристрою")
            print("8. Генерація звіту доступу")
            print("9. Сканування мережі")
            print("10. Вихід з системи")
            choice = input("Введіть ваш вибір: ")
            if choice == "1":
                self.register_user()
            elif choice == "2":
                self.add_network_device()
            elif choice == "3":
                self.add_device_configuration()
            elif choice == "4":
                self.assign_user_permission()
            elif choice == "5":
                self.list_users()
            elif choice == "6":
                self.list_devices()
            elif choice == "7":
                self.simulate_device_access()
            elif choice == "8":
                self.generate_access_report()
            elif choice == "9":
                self.network_device_scan()
            elif choice == "10":
                self.current_user = None
                break
            else:
                print("Невірний вибір. Будь ласка, оберіть правильну опцію.")

    def user_menu(self):
        #Меню для звичайних користувачів з обмеженим набором функцій
        while True:
            print("\nМеню користувача:")
            print("1. Виведення списку мережних пристроїв")
            print("2. Симуляція доступу до пристрою")
            print("3. Сканування мережі")
            print("4. Вихід з системи")
            choice = input("Введіть ваш вибір: ")
            if choice == "1":
                self.list_devices()
            elif choice == "2":
                self.simulate_device_access()
            elif choice == "3":
                self.network_device_scan()
            elif choice == "4":
                self.current_user = None
                break
            else:
                print("Невірний вибір. Будь ласка, спробуйте ще раз.")

    def start(self):
        #Запуск інтерфейсу системи адміністрування
        while True:
            print("\nЛаскаво просимо до системи адміністрування доступу до
мережних пристроїв")
            print("1. Вхід в систему")
            print("2. Вихід")
            choice = input("Введіть ваш вибір: ")
            if choice == "1":
                self.login()
                if self.current_user:
                    if self.current_user[3] == "admin":
                        self.admin_menu()
                    else:
                        self.user_menu()
```

КОПІЯ 2025

```
            elif choice == "2":
                print("Вихід з системи. До побачення!")
                break
            else:
                print("Невірний вибір. Будь ласка, спробуйте ще раз.")

#Функція для створення резервної копії бази даних
def backup_database(db_file):
    #Створення резервної копії файлу бази даних
    backup_file = db_file + ".bak"
    try:
        with open(db_file, "rb") as original:
            with open(backup_file, "wb") as backup:
                backup.write(original.read())
        print("Резервну копію створено в", backup_file)
    except Exception as e:
        print("Помилка створення резервної копії:", e)

#Функція для відновлення бази даних з резервної копії
def restore_database(db_file):
    #Відновлення бази даних з файлу резервної копії
    backup_file = db_file + ".bak"
    try:
        if os.path.exists(backup_file):
            with open(backup_file, "rb") as backup:
                with open(db_file, "wb") as original:
                    original.write(backup.read())
            print("Базу даних відновлено з", backup_file)
        else:
            print("Файл резервної копії не знайдено.")
    except Exception as e:
        print("Помилка відновлення бази даних:", e)

#Функція для виконання розширених запитів до бази даних
def perform_extended_queries(db):
    #Виконання ряду додаткових запитів для демонстрації складності системи
    print("Виконання розширених запитів...")
    users = db.get_users()
    for user in users:
        print("Користувач:", user[1], "Роль:", user[3])
    devices = db.get_devices()
    for device in devices:
        print("Пристрій:", device[1], "IP:", device[2])
    logs = db.get_access_logs()
    for log in logs:
        print("Запис логу:", log)
    permissions = db.get_user_permissions()
    for perm in permissions:
        print("Запис прав доступу:", perm)
    configs = db.get_device_configs()
    for config in configs:
        print("Конфігурація пристрою:", config)
    print("Розширені запити виконано.")

#Функція для симуляції детальної обробки даних
def verbose_processing():
    #Запуск симуляції детальної обробки даних
    print("Запуск детальної обробки даних...")
    for i in range(5):
        print("Крок обробки", i+1)
        time.sleep(0.2)
    print("Детальна обробка даних завершена.")

#Додаткова dummy-функція для розширення кількості рядків коду
def dummy_function_one():
    #Dummy-функція 1 для додаткових рядків коду
    for i in range(3):
        print("Dummy-функція 1, ітерація", i)
        time.sleep(0.1)
```

КПЗ - 2025

```
#Додаткова dummy-функція для розширення кількості рядків коду
def dummy_function_two():
    #Dummy-функція 2 для додаткових рядків коду
    for j in range(3):
        print("Dummy-функція 2, ітерація", j)
        time.sleep(0.1)

#Ще одна dummy-функція для збільшення обсягу коду
def dummy_function_three():
    #Dummy-функція 3 для додаткової обробки
    for k in range(3):
        print("Dummy-функція 3, ітерація", k)
        time.sleep(0.1)

#Ще одна dummy-функція для розширення коду
def dummy_function_four():
    #Dummy-функція 4 для додаткових рядків коду
    for l in range(3):
        print("Dummy-функція 4, ітерація", l)
        time.sleep(0.1)

#Функція для демонстрації індуського стилю розширення коду
def indian_style_extension():
    #Функція демонструє індуський стиль розширення кількості рядків коду
    for m in range(5):
        print("Індуський стиль розширення, рядок", m)
        time.sleep(0.1)

#Головна функція для запуску системи адміністрування
def main():
    #Ініціалізація системи адміністрування доступу
    admin_system = NetworkAccessAdmin()
    #Запуск детальної обробки даних для симуляції ініціалізації системи
    verbose_processing()
    #Запуск основного інтерфейсу системи
    admin_system.start()
    #Після виходу зі системи виконання розширених запитів до бази даних
    perform_extended_queries(admin_system.db)
    #Створення резервної копії бази даних
    backup_database("network_access.db")
    #Додаткова детальна обробка даних після створення резервної копії
    verbose_processing()
    #Виклик додаткових dummy-функцій для розширення коду
    dummy_function_one()
    dummy_function_two()
    dummy_function_three()
    dummy_function_four()
    #Демонстрація індуського стилю розширення коду
    indian_style_extension()

#Точка входу в програму
if __name__ == "__main__":
    #Запуск головної функції
    main()
```

КОПІЯ 2025

Файл integration_test_suite_extension.py

```
#!/usr/bin/env python3
# integration_test_suite_extension.py
# Набір інтеграційних тестів для перевірки сумісності всіх модулів системи

import time
import random
import json
import os

# Клас для управління інтеграційними тестами
class IntegrationTestSuite:
    def __init__(self):
        # Ініціалізація списку тестових сценаріїв
        self.test_cases = []
        self.results = []
        self.load_tests()

    def load_tests(self):
        # Завантаження тестових сценаріїв з dummy даних
        for i in range(10):
            test = {
                'name': f"Тестова ситуація {i+1}",
                'module': f"Модуль_{i+1}",
                'expected': random.choice(["успіх", "провал"]),
                'data': [random.randint(0, 100) for _ in range(20)]
            }
            self.test_cases.append(test)
        # Додаткове завантаження тестових сценаріїв
        for j in range(5):
            test = {
                'name': f"Додатковий тест {j+1}",
                'module': f"Додатковий_Модуль_{j+1}",
                'expected': random.choice(["успіх", "провал"]),
                'data': [random.randint(100, 200) for _ in range(30)]
            }
            self.test_cases.append(test)

    def run_test(self, test):
        # Виконання окремого тесту
        print(f"Запуск тесту: {test['name']} для {test['module']}")
        time.sleep(0.1)
        result = random.choice(["успіх", "провал"])
        test_result = {
            'name': test['name'],
            'module': test['module'],
            'expected': test['expected'],
            'result': result,
            'data': test['data']
        }
        print(f"Результат тесту: {result}")
        return test_result
```

КПІВ_2025

```
    def run_all_tests(self):
        # Запуск всіх тестових сценаріїв
        print("Початок виконання інтеграційних тестів...")
        for test in self.test_cases:
            result = self.run_test(test)
            self.results.append(result)
            time.sleep(0.05)
        print("Всі інтеграційні тести виконано.")
        self.generate_report()

    def generate_report(self):
        # Генерація звіту за результатами тестів
        print("Генерація звіту за результатами інтеграційних тестів...")
        success_count = 0
        fail_count = 0
        for res in self.results:
            if res['result'] == "успіх":
                success_count += 1
            else:
                fail_count += 1
        report = {
            'total_tests': len(self.results),
            'успіх': success_count,
            'провал': fail_count,
            'details': self.results
        }
        report_file = "integration_test_report.json"
        with open(report_file, "w", encoding="utf-8") as f:
            json.dump(report, f, ensure_ascii=False, indent=4)
        print(f"Звіт збережено у файл: {report_file}")

    def detailed_logging(self):
        # Детальне логування кожного кроку тестування
        print("Детальне логування тестів...")
        for index, test in enumerate(self.test_cases):
            log_message = f"Тест {index+1}: {test['name']} - дані:
{test['data']}"
            print(log_message)
            time.sleep(0.02)
        print("Детальне логування завершено.")

# Додаткова функція для симуляції складних тестів
def simulate_complex_scenario():
    # Симуляція складного сценарію з численними кроками
    print("Симуляція складного сценарію інтеграційного тестування почалася...")
    steps = 15
    for step in range(steps):
        print(f"Крок {step+1} з {steps}: Виконання операції...")
        for inner in range(5):
            print(f"  Підкрок {inner+1}: Обробка даних {random.randint(1,
100)}")
            time.sleep(0.03)
```

КОПЗ - 2025

```
            print("  Завершення кроку.")
            time.sleep(0.1)
        print("Симуляція складного сценарію завершена.")

def main():
    # Головна функція для запуску інтеграційних тестів
    print("Запуск інтеграційного тестового набору...")
    test_suite = IntegrationTestSuite()
    test_suite.detailed_logging()
    simulate_complex_scenario()
    test_suite.run_all_tests()
    print("Інтеграційне тестування завершено.")

if __name__ == "__main__":
    main()
```

КбПЗ_2025

Файл simulation_environment_extension.py

```
#!/usr/bin/env python3
# simulation_environment_extension.py
# Створення симуляційного середовища для імітації роботи мережі

import time
import random

# Клас для симуляції мережного пристрою
class SimulatedDevice:
    def __init__(self, device_id, device_name, ip_address):
        # Ініціалізація параметрів пристрою
        self.device_id = device_id
        self.device_name = device_name
        self.ip_address = ip_address
        self.status = "невідомо"
        self.load = 0

    def update_status(self):
        # Випадкова зміна статусу пристрою
        self.status = random.choice(["онлайн", "офлайн", "перезавантаження"])
        self.load = random.randint(0, 100)
        print(f"Пристрій {self.device_name} ({self.ip_address}) оновив статус:
{self.status} з навантаженням {self.load}%")
        time.sleep(0.1)

    def simulate_activity(self):
        # Симуляція активності пристрою
        print(f"Симуляція активності пристрою {self.device_name}...")
        for i in range(3):
            activity = random.choice(["обробка запитів", "передача даних",
"очікування"])
            print(f"  Активність {i+1}: {activity}")
            time.sleep(0.1)

# Клас для симуляції мережного середовища
class SimulationEnvironment:
    def __init__(self):
        # Ініціалізація списку пристроїв
        self.devices = []
        self.simulation_time = 0
        self.generate_devices()

    def generate_devices(self):
        # Генерація dummy мережних пристроїв
        print("Генерація мережних пристроїв для симуляції...")
        for i in range(20):
            device = SimulatedDevice(
                device_id=i+1,
                device_name=f"Пристрій_{i+1}",
                ip_address=f"192.168.1.{i+1}"
            )
```

КбТз_2025

```
            self.devices.append(device)
            print(f"  Згенеровано: {device.device_name} з IP
{device.ip_address}")
            time.sleep(0.05)
        print("Генерація пристроїв завершена.")

    def run_simulation(self, duration_seconds):
        # Запуск симуляції мережного середовища на заданий час
        print("Запуск симуляції мережного середовища...")
        start_time = time.time()
        while time.time() - start_time < duration_seconds:
            self.simulation_time += 1
            print(f"Симуляційний час: {self.simulation_time} секунда(и)")
            for device in self.devices:
                device.update_status()
                device.simulate_activity()
            print("-" * 40)
            time.sleep(0.5)
        print("Симуляція завершена.")

    def simulate_network_events(self):
        # Симуляція випадкових мережних подій
        print("Симуляція випадкових мережних подій...")
        events = ["атака", "відключення", "автоматичне відновлення", "оновлення
програмного забезпечення"]
        for i in range(10):
            event = random.choice(events)
            affected_device = random.choice(self.devices)
            print(f"Подія {i+1}: {event} на {affected_device.device_name}
({affected_device.ip_address})")
            time.sleep(0.2)
        print("Симуляція мережних подій завершена.")

def main():
    # Головна функція для запуску симуляційного середовища
    print("Ініціалізація симуляційного середовища мережі...")
    env = SimulationEnvironment()
    env.simulate_network_events()
    env.run_simulation(5)
    print("Симуляційне середовище завершило роботу.")

if __name__ == "__main__":
    main()
```

КОТЗ 2025

Файл network_topology_mapping_extension.py

```
#!/usr/bin/env python3
# network_topology_mapping_extension.py
# Створення карти мережевої топології та генерація текстових схем топології

import time
import random

# Клас для представлення мережевого вузла
class NetworkNode:
    def __init__(self, node_id, node_name):
        # Ініціалізація параметрів вузла
        self.node_id = node_id
        self.node_name = node_name
        self.connections = []

    def add_connection(self, other_node):
        # Додавання зв'язку з іншим вузлом
        if other_node not in self.connections:
            self.connections.append(other_node)
            print(f"Зв'язок додано: {self.node_name} -> {other_node.node_name}")

# Клас для управління мережею топології
class NetworkTopology:
    def __init__(self):
        # Ініціалізація списку вузлів
        self.nodes = []
        self.generate_nodes()
        self.generate_random_connections()

    def generate_nodes(self):
        # Генерація dummy вузлів мережі
        print("Генерація вузлів мережевої топології...")
        for i in range(15):
            node = NetworkNode(node_id=i+1, node_name=f"Вузол_{i+1}")
            self.nodes.append(node)
            print(f"  Створено: {node.node_name}")
            time.sleep(0.05)
        print("Генерація вузлів завершена.")

    def generate_random_connections(self):
        # Генерація випадкових з'єднань між вузлами
        print("Генерація випадкових з'єднань між вузлами...")
        for node in self.nodes:
            connection_count = random.randint(1, 4)
            for _ in range(connection_count):
                target_node = random.choice(self.nodes)
                if target_node != node:
                    node.add_connection(target_node)
                    time.sleep(0.02)
        print("Генерація з'єднань завершена.")
```

КБПЗ_2025

```
    def display_topology(self):
        # Відображення мережевої топології у вигляді текстової схеми
        print("Відображення мережевої топології:")
        for node in self.nodes:
            connections = ", ".join([conn.node_name for conn in
node.connections])
            print(f"{node.node_name} -> {connections}")
            time.sleep(0.05)
        print("Відображення топології завершено.")

    def generate_ascii_diagram(self):
        # Генерація ASCII діаграми для мережевої топології
        print("Генерація ASCII діаграми мережевої топології...")
        diagram = ""
        for node in self.nodes:
            diagram += f"[{node.node_name}]"
            if node.connections:
                diagram += " -- " + " | ".join([f"[{conn.node_name}]" for conn
in node.connections])
            diagram += "\n"
            time.sleep(0.05)
        print(diagram)
        print("ASCII діаграма згенерована.")

def main():
    # Головна функція для запуску генерації топології
    print("Ініціалізація системи генерації мережевої топології...")
    topology = NetworkTopology()
    topology.display_topology()
    topology.generate_ascii_diagram()
    print("Генерація мережевої топології завершена.")

if __name__ == "__main__":
    main()
```

КОПЗ_2025

Файл performance_benchmark_extension.py

```
#!/usr/bin/env python3
# performance_benchmark_extension.py
# Проведення бенчмаркінгу системи, вимірювання швидкості виконання завдань та
аналіз продуктивності

import time
import random
import statistics

# Функція для бенчмаркінгу сортування великих списків
def benchmark_sorting():
    print("Бенчмаркінг сортування великих списків...")
    data_sizes = [1000, 5000, 10000, 20000, 50000]
    results = {}
    for size in data_sizes:
        data = [random.randint(0, 1000000) for _ in range(size)]
        start_time = time.time()
        sorted_data = sorted(data)
        end_time = time.time()
        duration = end_time - start_time
        results[size] = duration
        print(f"  Сортування списку з {size} елементів зайняло {duration:.4f}
секунд")
        time.sleep(0.1)
    return results

# Функція для бенчмаркінгу обчислювальних задач
def benchmark_computation():
    print("Бенчмаркінг обчислювальних задач...")
    iterations = 1000000
    start_time = time.time()
    total = 0
    for i in range(iterations):
        total += (i % 7) * (i % 5)
        if i % 200000 == 0:
            print(f"  Обчислено {i} ітерацій...")
    end_time = time.time()
    duration = end_time - start_time
    print(f"  Обчислення за {iterations} ітерацій зайняло {duration:.4f}
секунд")
    return duration

# Функція для тестування продуктивності операцій над масивами
def benchmark_array_operations():
    print("Бенчмаркінг операцій над масивами...")
    array_size = 100000
    data = [random.random() for _ in range(array_size)]
    start_time = time.time()
    squared = [x * x for x in data]
    sum_value = sum(squared)
    end_time = time.time()
```

КОПЗ_2025

```
    duration = end_time - start_time
    print(f"  Операції над масивом з {array_size} елементів зайняли
{duration:.4f} секунд")
    return duration

# Функція для генерації детального звіту з бенчмаркінгу
def generate_benchmark_report():
    print("Генерація звіту з бенчмаркінгу продуктивності...")
    sort_results = benchmark_sorting()
    computation_time = benchmark_computation()
    array_op_time = benchmark_array_operations()

    durations = list(sort_results.values()) + [computation_time, array_op_time]
    average_time = statistics.mean(durations)
    report = {
        'sort_results': sort_results,
        'computation_time': computation_time,
        'array_operation_time': array_op_time,
        'average_time': average_time,
        'total_tests': len(durations)
    }

    print("Детальний звіт бенчмаркінгу:")
    print("  Середній час виконання:", average_time)
    print("  Загальна кількість тестів:", len(durations))
    for key, value in report.items():
        print(f"  {key}: {value}")
    print("Звіт з бенчмаркінгу завершено.")

def main():
    # Головна функція для запуску бенчмаркінгу
    print("Запуск тестів бенчмаркінгу продуктивності системи...")
    generate_benchmark_report()
    print("Бенчмаркінг завершено.")

if __name__ == "__main__":
    main()
```

КОПЗ – 2025

Файл plugin_manager_extension.py

```
#!/usr/bin/env python3
# plugin_manager_extension.py
# Менеджер плагінів для системи, що дозволяє підключати, оновлювати та управляти
додатковими модулями розширення

import os
import sys
import time
import importlib.util

# Клас для управління плагінами
class PluginManager:
    def __init__(self, plugins_directory="plugins"):
        # Ініціалізація директорії плагінів та списку завантажених плагінів
        self.plugins_directory = plugins_directory
        self.plugins = {}
        self.load_plugins()

    def load_plugins(self):
        # Завантаження плагінів з вказаної директорії
        print(f"Пошук плагінів у директорії: {self.plugins_directory}")
        if not os.path.exists(self.plugins_directory):
            print("Директорія плагінів не існує. Створення директорії...")
            os.makedirs(self.plugins_directory)
        plugin_files = [f for f in os.listdir(self.plugins_directory) if
f.endswith(".py")]
        print(f"Знайдено {len(plugin_files)} файлів плагінів.")
        for plugin_file in plugin_files:
            plugin_path = os.path.join(self.plugins_directory, plugin_file)
            plugin_name = os.path.splitext(plugin_file)[0]
            self.load_plugin(plugin_name, plugin_path)
            time.sleep(0.1)

    def load_plugin(self, plugin_name, plugin_path):
        # Завантаження окремого плагіна
        print(f"Завантаження плагіна: {plugin_name} з файлу {plugin_path}")
        spec = importlib.util.spec_from_file_location(plugin_name, plugin_path)
        if spec and spec.loader:
            module = importlib.util.module_from_spec(spec)
            try:
                spec.loader.exec_module(module)
                self.plugins[plugin_name] = module
                print(f"Плагін {plugin_name} успішно завантажено.")
            except Exception as e:
                print(f"Помилка завантаження плагіна {plugin_name}: {e}")
        else:
            print(f"Не вдалося створити spec для плагіна {plugin_name}.")

    def list_plugins(self):
        # Виведення списку завантажених плагінів
        print("Список завантажених плагінів:")
```

КПЗ-2025

```
        for name in self.plugins.keys():
            print(f"  - {name}")
        if not self.plugins:
            print("Плагіни не завантажено.")

    def execute_plugin_function(self, plugin_name, function_name,
                                *args, **kwargs):
        # Виконання функції плагіна з переданими параметрами
        if plugin_name in self.plugins:
            plugin = self.plugins[plugin_name]
            func = getattr(plugin, function_name, None)
            if callable(func):
                print(f"Виконання функції {function_name} плагіна
                      {plugin_name}...")
                try:
                    result = func(*args, **kwargs)
                    print(f"Результат виконання: {result}")
                except Exception as e:
                    print(f"Помилка виконання функції {function_name} плагіна
                          {plugin_name}: {e}")
            else:
                print(f"Функція {function_name} не знайдена в плагіні
                     {plugin_name}.")
        else:
            print(f"Плагін {plugin_name} не завантажено.")

    def reload_plugins(self):
        # Перезавантаження всіх плагінів
        print("Перезавантаження всіх плагінів...")
        for plugin_name in list(self.plugins.keys()):
     plugin_path = os.path.join(self.plugins_directory, plugin_name + ".py")
     self.load_plugin(plugin_name, plugin_path)
        print("Перезавантаження плагінів завершено.")

def main():
    # Головна функція для менеджера плагінів
    print("Ініціалізація менеджера плагінів...")
    manager = PluginManager()
    manager.list_plugins()
    # Демонстрація виконання функцій з завантажених плагінів
    for plugin_name in manager.plugins.keys():
        manager.execute_plugin_function(plugin_name, "dummy_plugin_function",
                                        "Тестовий параметр")
    # Перезавантаження плагінів для перевірки оновлень
    manager.reload_plugins()
    print("Менеджер плагінів завершив роботу.")

if __name__ == "__main__":
    main()
```

КОПЗ - 2025

Файл user_auth_extension.py

```
#!/usr/bin/env python3
# user_auth_extension.py
# Розширене управління аутентифікацією користувачів із підтримкою
багатофакторної перевірки,
# генерації OTP-кодів та симуляції біометричної аутентифікації.

import time
import random
import hashlib

# Клас для розширеної аутентифікації користувачів
class ExtendedUserAuth:
    def __init__(self):
        # Ініціалізація бази даних користувачів (dummy)
        self.users = {}  # username: {password_hash: ..., otp: ..., biometric:
...}
        self.otp_validity = 300  # час дії OTP в секундах
        self.otp_storage = {}  # username: (otp, timestamp)

    def hash_password(self, password):
        # Генерація хешу пароля
        return hashlib.sha256(password.encode()).hexdigest()

    def register_user(self, username, password):
        # Реєстрація нового користувача
        if username in self.users:
            print("Користувач вже існує.")
            return False
        password_hash = self.hash_password(password)
        self.users[username] = {'password_hash': password_hash,
                                'biometric': None}
        print(f"Користувача {username} зареєстровано.")
        return True

    def login(self, username, password):
        # Вхід користувача за допомогою пароля
        if username not in self.users:
            print("Користувача не знайдено.")
            return False
        password_hash = self.hash_password(password)
        if self.users[username]['password_hash'] != password_hash:
            print("Невірний пароль.")
            return False
        print("Пароль підтверджено. Починається багатофакторна аутентифікація.")
        return self.multi_factor_auth(username)

    def generate_otp(self, username):
        # Генерація OTP-коду
        otp = random.randint(100000, 999999)
        timestamp = time.time()
        self.otp_storage[username] = (otp, timestamp)
```

КбПЗ_2025

```
        print(f"OTP для {username}: {otp} (діє 300 секунд)")
        return otp

    def verify_otp(self, username, input_otp):
        # Перевірка OTP-коду
        if username not in self.otp_storage:
            print("OTP не згенеровано.")
            return False
        otp, timestamp = self.otp_storage[username]
        current_time = time.time()
        if current_time - timestamp > self.otp_validity:
            print("OTP закінчився.")
            return False
        if int(input_otp) == otp:
            print("OTP успішно підтверджено.")
            return True
        else:
            print("Невірний OTP.")
            return False

    def biometric_auth(self, username):
        # Симуляція біометричної аутентифікації
        print("Сканування біометричних даних для", username)
        time.sleep(1)
        # Використовуємо dummy перевірку: 80% шанс успіху
        result = random.random() < 0.8
        if result:
            print("Біометрична аутентифікація успішна.")
        else:
            print("Біометрична аутентифікація не вдалася.")
        return result

    def multi_factor_auth(self, username):
        # Багатофакторна аутентифікація: спочатку OTP, потім біометрія
        self.generate_otp(username)
        user_otp = input("Введіть отриманий OTP: ")
        if not self.verify_otp(username, user_otp):
            print("Багатофакторна аутентифікація не пройдена (OTP).")
            return False
        if not self.biometric_auth(username):
            print("Багатофакторна аутентифікація не пройдена (біометрія).")
            return False
        print("Багатофакторна аутентифікація успішно завершена. Доступ надано.")
        return True

    def simulate_login_flow(self):
        # Симуляція повного процесу реєстрації та входу користувача
        print("Симуляція аутентифікації користувача починається...")
        username = input("Введіть ім'я користувача для реєстрації: ")
        password = input("Введіть пароль для реєстрації: ")
        self.register_user(username, password)
        print("Спроба входу в систему:")
        login_password = input("Введіть пароль для входу: ")
```

КПЗ – 2025

```
        success = self.login(username, login_password)
        if success:
            print("Користувач успішно увійшов в систему.")
        else:
            print("Вхід в систему не вдався.")
        print("Симуляція аутентифікації завершена.")

def main():
    # Головна функція для тестування розширеної аутентифікації
    auth_system = ExtendedUserAuth()
    auth_system.simulate_login_flow()
    # Додаткове тестування реєстрації декількох користувачів
    for i in range(3):
        uname = f"user_{i+1}"
        pwd = f"pass_{i+1}"
        auth_system.register_user(uname, pwd)
        time.sleep(0.2)
    print("Список зареєстрованих користувачів:")
    for user in auth_system.users:
        print(f"  - {user}")
    print("Завершення модуля аутентифікації.")

if __name__ == "__main__":
    main()
```

КбПЗ_2025

Файл security_audit_extension.py

```
#!/usr/bin/env python3
# security_audit_extension.py
# Проведення розширених аудитів безпеки, аналіз доступу
# та виявлення аномалій у системі

import time
import random
import json

# Клас для проведення аудиту безпеки
class SecurityAudit:
    def __init__(self):
        # Ініціалізація dummy логів та даних аудиту
        self.access_logs = []
        self.audit_events = []
        self.generate_dummy_logs()

    def generate_dummy_logs(self):
        # Генерація dummy логів доступу
        print("Генерація dummy логів доступу для аудиту...")
        for i in range(50):
            log = {
                'event_id': i+1,
                'user': f"user_{random.randint(1,10)}",
                'action': random.choice(["login", "logout", "access_device",
"config_change"]),
                'status': random.choice(["успіх", "провал"]),
                'timestamp': time.time() - random.randint(0, 10000)
            }
            self.access_logs.append(log)
            time.sleep(0.01)
        print("Генерація логів завершена.")

    def detect_anomalies(self):
        # Виявлення аномалій у логах доступу
        print("Аналіз логів для виявлення аномалій...")
        anomalies = []
        for log in self.access_logs:
            # Dummy логіка: якщо статус "провал" та випадкове число > 0.5
            if log['status'] == "провал" and random.random() > 0.5:
                anomalies.append(log)
                print(f"Аномалія виявлена: {log}")
            time.sleep(0.005)
        print(f"Знайдено {len(anomalies)} аномалій.")
        return anomalies

    def generate_audit_report(self):
        # Генерація детального звіту аудиту
        print("Генерація звіту аудиту безпеки...")
        anomalies = self.detect_anomalies()
        report = {
```

КБПВ-2025

```
            'total_logs': len(self.access_logs),
            'anomalies_found': len(anomalies),
            'anomaly_details': anomalies,
            'audit_time': time.ctime()
        }
        report_filename = "security_audit_report.json"
        with open(report_filename, "w", encoding="utf-8") as f:
            json.dump(report, f, ensure_ascii=False, indent=4)
        print(f"Звіт аудиту збережено у файл: {report_filename}")

    def simulate_audit(self):
        # Симуляція повного процесу аудиту
        print("Початок аудиту безпеки...")
        time.sleep(0.5)
        self.generate_audit_report()
        print("Аудит безпеки завершено.")

def main():
    # Головна функція для запуску аудиту безпеки
    audit = SecurityAudit()
    audit.simulate_audit()
    # Додаткова симуляція подій аудиту
    for i in range(5):
        event = {
            'event': f"Незвичайна активність {i+1}",
            'detail': f"Детальна інформація про подію {i+1}",
            'timestamp': time.ctime()
        }
        audit.audit_events.append(event)
        print(f"Зафіксовано подію: {event}")
        time.sleep(0.1)
    print("Список додаткових подій аудиту:")
    for event in audit.audit_events:
        print(event)
    print("Завершення модуля аудиту безпеки.")

if __name__ == "__main__":
    main()
```

КОПЗ_2025