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
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти

на тему

“ Дослідження та програмна реалізація системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS ”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-23М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Прокопчук Р.С.
«____» _____ 2024 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

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
« » _____2024 року

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

Прокопчук Роман Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 07.08.2024 року

3. Строк подання студентом роботи до захисту *02.12.2024р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження механізмів автентифікації та розробка ефективної системи захисту доступу до інтернет-ресурсів на основі FreeRADIUS*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Маркетингове та економічне обґрунтування ІТ проекту.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Доренська А.О.	05.10.2024	14.11.2024
Охорона праці	Марченко К.М.	06.10.2024	16.11.2024

7. Дата видачі завдання « » 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2024р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2024р.	
3.	Розробка моделі компонента	20.10.2024р.	
4.	Розробка структур даних	25.10.2024р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2024р.	
6.	Програмування алгоритмів	10.11.2024р.	
7.	Розрахунок економічної ефективності	13.11.2024р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2024р.	
9.	Оформлення ПЗ	17.11.2024р.	
10.	Попередній захист роботи	02.12.2024р.	

Дата видачі завдання
« » 2024р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« » 2024 р.

Підпис здобувача
_____ Прокопчук Р.С.
(прізвище та ініціали)

АНОТАЦІЯ

Прокопчук Р.С. Дослідження та програмна реалізація системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

У даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення для реалізації системи автентифікації користувачів в інтернет-мережах з використанням сервера FreeRADIUS.

Метою роботи є дослідження та впровадження сучасних методів автентифікації, інтеграція FreeRADIUS у середовища різного рівня складності та підвищення безпеки доступу до інтернет-ресурсів.

Об'єктом дослідження є сервер автентифікації FreeRADIUS, його модулі, можливості інтеграції та існуючі протоколи взаємодії з іншими мережевими компонентами.

Результат роботи – створено комплексну систему автентифікації користувачів, яка може застосовуватись у корпоративних і публічних мережах для забезпечення захищеного доступу до інтернет-ресурсів. Розроблене рішення включає налаштування FreeRADIUS, адаптацію під конкретні сценарії використання, а також рекомендації щодо масштабування та оптимізації роботи сервера.

В ході виконання магістерської роботи:

- проаналізовано методи автентифікації, які підтримуються FreeRADIUS;

- проведено інтеграцію сервера з іншими компонентами мережі, зокрема базами даних MySQL та LDAP;

- налаштовано підтримку протоколів EAP-TLS, EAP-TTLS, та PEAP.

Ключові слова: FreeRADIUS, автентифікація, EAP, безпека мереж, протокол RADIUS, сервер доступу.

ABSTRACT

Prokopchuk R.S. Research and software implementation of a user authentication system in Internet networks based on FreeRADIUS. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the second (master's) level of higher education, software is developed for implementing a user authentication system in Internet networks using the FreeRADIUS server.

The purpose of the work is to research and implement modern authentication methods, integrate FreeRADIUS into various network environments, and enhance the security of access to Internet resources.

The object of the study is the FreeRADIUS authentication server, its modules, integration capabilities, and existing interaction protocols with other network components.

The result of the work is the creation of a comprehensive user authentication system that can be used in corporate and public networks to ensure secure access to Internet resources. The developed solution includes configuring FreeRADIUS, adapting it to specific usage scenarios, and providing recommendations for scaling and optimizing server performance.

In the course of this master's thesis:

- methods of authentication supported by FreeRADIUS were analyzed;

- the integration of the server with other network components, including MySQL and LDAP databases, was performed;

- support for EAP-TLS, EAP-TTLS, and PEAP protocols was configured;

Keywords: FreeRADIUS, authentication, EAP, network security, RADIUS protocol, access server.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

RADIUS	–	Remote Authentication Dial-In User ServiceModel
VPN	–	Virtual Private Network
ISE	–	Identity Services Engine
БД	–	База даних
NPS	–	Network Policy Server
EAP-TTLS	–	Extensible Authentication Protocol - Tunneled
	–	Transport Layer Security.
EAP-PEAP	–	Extensible Authentication Protocol - Protected
	–	Extensible Authentication Protocol
SQL	–	Structured Query Language.
NOSQL	–	Non-relational Structured Query Language
	–	
	–	
	–	
	–	
	–	
	–	
	–	
	–	
	–	
	–	

ВСТУП

Актуальність теми. Забезпечення безпеки доступу до інтернет-мереж набуло критичного значення в умовах сучасного інформаційного суспільства, де велика кількість даних передається та обробляється через мережеві інфраструктури. З кожним роком зростають вимоги до систем автентифікації, які дозволяють контролювати доступ до мережевих ресурсів та забезпечувати конфіденційність даних. Надійна система автентифікації є основою інформаційної безпеки, оскільки вона дозволяє захищати інформацію від несанкціонованого доступу, ідентифікувати користувачів та відслідковувати їх дії.

Одна з найбільш розповсюджених технологій для організації доступу до мережі - це протокол RADIUS, який забезпечує централізовану автентифікацію, авторизацію та облік користувачів.

У рамках цієї роботи буде досліджено можливості FreeRADIUS для побудови надійної системи автентифікації користувачів в інтернет-мережах, визначено основні підходи до інтеграції цього рішення з іншими елементами інфраструктури та розроблено програмну реалізацію системи. Буде детально проаналізовано процеси автентифікації за допомогою FreeRADIUS та переваги використання цього інструменту у корпоративних, освітніх та інших середовищах з великою кількістю користувачів.

Мета й завдання дослідження. Метою роботи є дослідження механізмів автентифікації та розробка ефективної системи захисту доступу до інтернет-ресурсів на основі FreeRADIUS, що задовольняє вимоги сучасних мереж щодо безпеки та надійності.

Для досягнення мети роботи поставлені наступні **задачі**:

- дослідити можливості FreeRADIUS;

- висвітлити кожний етап при створені системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS;

– дослідити альтернативні рішення для аутентифікації користувачів у мережах;

– розробити програмну реалізацію ситеми і провести її тестування.

Об'єктом дослідження є процеси автентифікації, авторизації та обліку (AAA) в сучасних мережах.

Предметом дослідження є сервер FreeRADIUS як інструмент реалізації систем централізованої автентифікації та управління доступом.

Методи дослідження включають:

- аналіз ефективності протоколів аутентифікації;

- дослідження методів інтеграції FreeRADIUS із зовнішніми базами даних (SQL та NoSQL);

- тестування продуктивності та навантаження на сервер автентифікації;

- дослідження методів забезпечення безпеки та захисту від атак.

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

1) Вперше запропоновано методологію інтеграції серверу FreeRADIUS з адаптивними алгоритмами автентифікації, які враховують поведінкові особливості користувачів та контекстні фактори доступу (наприклад, геолокацію, час доби, пристрій).

2) Розроблено вдосконалену архітектуру системи автентифікації, що забезпечує підвищену стійкість до атак методом підбору паролів та атак "людина посередині" (MITM), завдяки використанню сучасних протоколів (наприклад, EAP-TTLS або EAP-PEAP).

3) Удосконалено процедури логування та аудиту в FreeRADIUS, що дозволяє автоматично аналізувати події для виявлення потенційних загроз безпеці.

4) Проведено аналіз продуктивності FreeRADIUS в умовах високого навантаження на основі власного тестового стенда, що дозволило оптимізувати його конфігурацію для роботи в масштабованих інтернет-мережах.

5) Запропоновано алгоритм інтеграції FreeRADIUS із зовнішніми базами даних (SQL/NoSQL) для динамічного управління доступом, зокрема на основі політик і ролей.

Практична цінність отриманих результатів досягнена у вигляді автоматичності та повної автоматичності алгоритмів без використання застосунків чи втручання користувача.

Достовірність наукових результатів підтверджена теоретичними дослідження та математичними розрахунками, даними отриманими при тестуванні.

Робота апробована на Всеукраїнській науково-практичній on-line конференції “Проблеми енергоефективності та автоматизації в промисловості та сільському господарстві”, яка відбулася 13-14 листопада 2024 року у ЦНТУ м. Кропивницький.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система автентифікації користувачів на основі FreeRADIUS є рішенням для забезпечення безпеки доступу до інтернет-мереж. Вона дозволяє ефективно управляти доступом, зберігати дані про авторизацію та використовувати різні методи автентифікації. FreeRADIUS підтримує низку сучасних протоколів, таких як RADIUS, що дозволяє інтегрувати систему з іншими інфраструктурами та платформами. Вона широко застосовується для забезпечення безпечного доступу до інтернет-ресурсів у великих корпоративних мережах, освітніх закладах та інших середовищах з великою кількістю користувачів.

1.2 Область застосування

FreeRADIUS є потужним інструментом для організації різних методів автентифікації на основі мережевих протоколів. Одним з основних протоколів, які підтримуються цією системою, є RADIUS, що забезпечує централізовану автентифікацію, авторизацію та облік. Це дозволяє використовувати систему для автентифікації у бездротових мережах, VPN, корпоративних мережах та інших середовищах, де потрібен контроль доступу до мережі.

Архітектура FreeRADIUS дозволяє здійснювати налаштування для різних типів автентифікації, таких як:

1) Проста автентифікація за допомогою імені користувача та пароля. Цей метод є основним і передбачає передачу даних автентифікації з клієнта на сервер у форматі RADIUS-пакетів.

2) Аналіз сертифікатів - для корпоративних середовищ або VPN підключень, де важлива додаткова безпека.

3) Двофакторна автентифікація. У таких випадках FreeRADIUS може інтегруватися з зовнішніми системами (наприклад, Google Authenticator) для реалізації багаторівневої автентифікації.

Основною перевагою є можливість інтеграції з різними базами, що забезпечує гнучкість та зручність управління автентифікацією. Це дозволяє використовувати систему у великих компаніях з численними підрозділами та ієрархічною структурою доступу, а також у навчальних закладах, де облік доступу потребує гнучкості.

Основна мета системи автентифікації на основі FreeRADIUS полягає у забезпеченні контролю доступу до мережі шляхом автентифікації користувачів і пристроїв. Це дозволяє:

Забезпечити конфіденційність та захист інформації. Автентифікація користувачів передбачає перевірку кожного користувача перед доступом до ресурсів мережі.

Зменшити ризики несанкціонованого доступу. Застосування FreeRADIUS дозволяє адміністраторам мережі обмежувати доступ, визначати рівні доступу до ресурсів та моніторити активність користувачів.

Надати можливість масштабування у великих мережах, таких як університетські та корпоративні мережі, що забезпечує можливість підтримки тисяч користувачів з різними рівнями доступу.

Організувати точний облік використання ресурсів. Використання облікової функції системи дозволяє адміністраторам відслідковувати використання ресурсів, оптимізувати їх розподіл та запобігати перевантаженням мережі.

Таким чином, система на основі FreeRADIUS є ключовим елементом для побудови безпечної інфраструктури автентифікації в інтернет-мережах. Вона надає адміністраторам можливість налаштувати захист від несанкціонованого доступу, обліковувати дії користувачів і підтримувати цілісність мережі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

У сучасному світі безпека та контроль доступу до інтернет-мереж є надзвичайно важливими аспектами, особливо у зв'язку з ростом кількості користувачів і послуг, які надаються через глобальну мережу. Одним із найпоширеніших рішень для аутентифікації користувачів є FreeRADIUS, який широко застосовується в телекомунікаціях та інших сферах для забезпечення доступу до мережевих ресурсів.

FreeRADIUS – це потужний сервер аутентифікації, який підтримує різні протоколи, такі як RADIUS, TACACS+ (Terminal Access Controller Access-Control System Plus) та Diameter[1]. Цей сервер забезпечує централізоване управління доступом до мережевих ресурсів, що дозволяє ефективно аутентифікувати користувачів на основі їх облікових даних, що зберігаються в базах даних, LDAP або Active Directory.

Крім FreeRADIUS, на ринку існують й інші рішення для аутентифікації, такі як Cisco Identity Services Engine (ISE) та Microsoft NPS. Cisco ISE пропонує розширені можливості управління доступом та безпекою, включаючи політики контролю доступу на основі ролей і автоматизацію процесів безпеки. Microsoft NPS, з іншого боку, інтегрується з Windows Server для управління доступом до ресурсів у локальних мережах та через VPN-з'єднання[7].

Cisco ISE - це рішення для управління ідентичністю та доступом, яке забезпечує централізовану аутентифікацію, авторизацію та облік (AAA) для пристроїв у корпоративних мережах. Воно дозволяє організаціям впроваджувати політики доступу на основі ролей, що підвищує рівень безпеки та контролю.

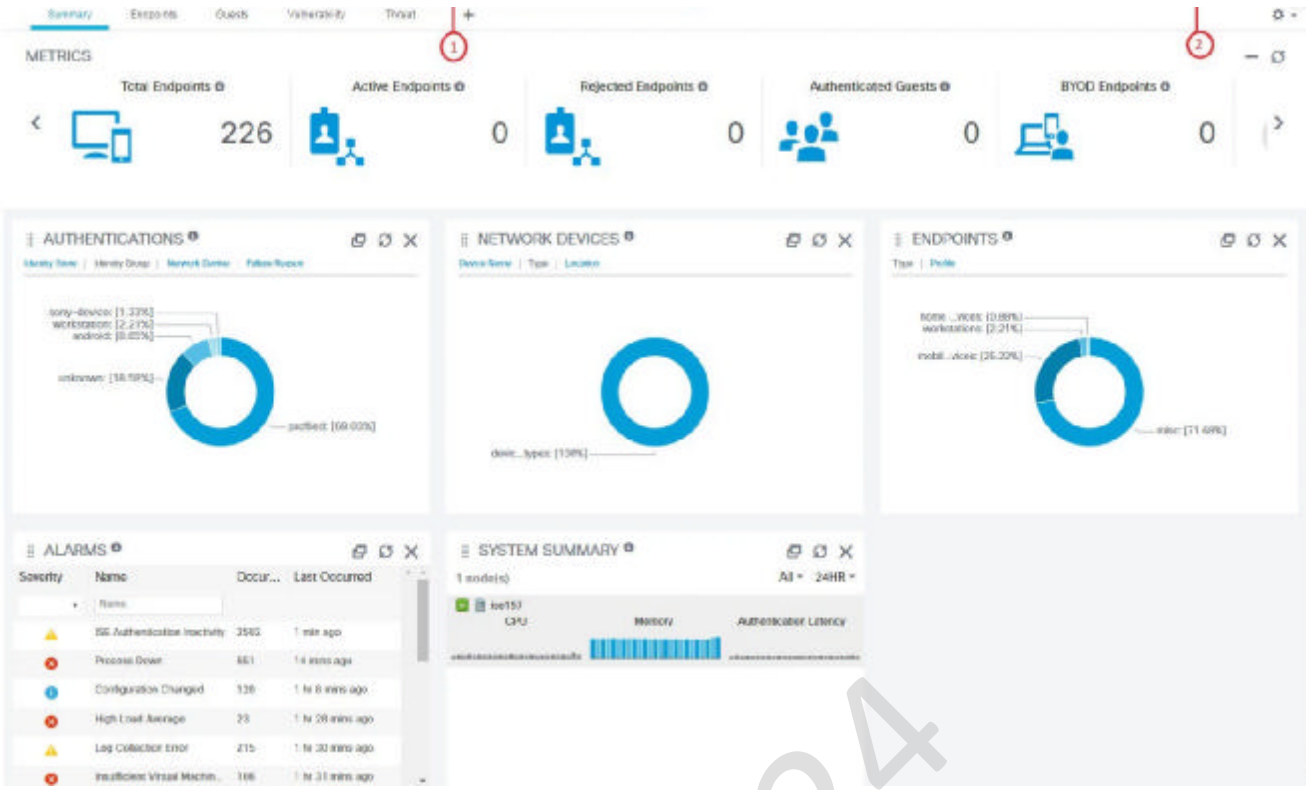

Рисунок 2.1 - Інтерфейс роботи програмного забезпечення Cisco Identity Services Engine

Розглянемо переваги:

1) централізоване управління: Cisco ISE дозволяє централізовано управляти доступом до мережі, що спрощує адміністрування та моніторинг;

2) гнучкість політик: Можливість налаштування політик доступу залежно від ролей користувачів і типів пристроїв;

3) багатофакторна автентифікація: Підтримка різних методів автентифікації для підвищення безпеки.

Недоліки:

1) вартість: Cisco ISE є досить дорогим рішенням, що може бути недоступним для малих підприємств;

2) складність налаштування: Впровадження та налаштування Cisco ISE можуть бути складними і вимагати часу та ресурсів.

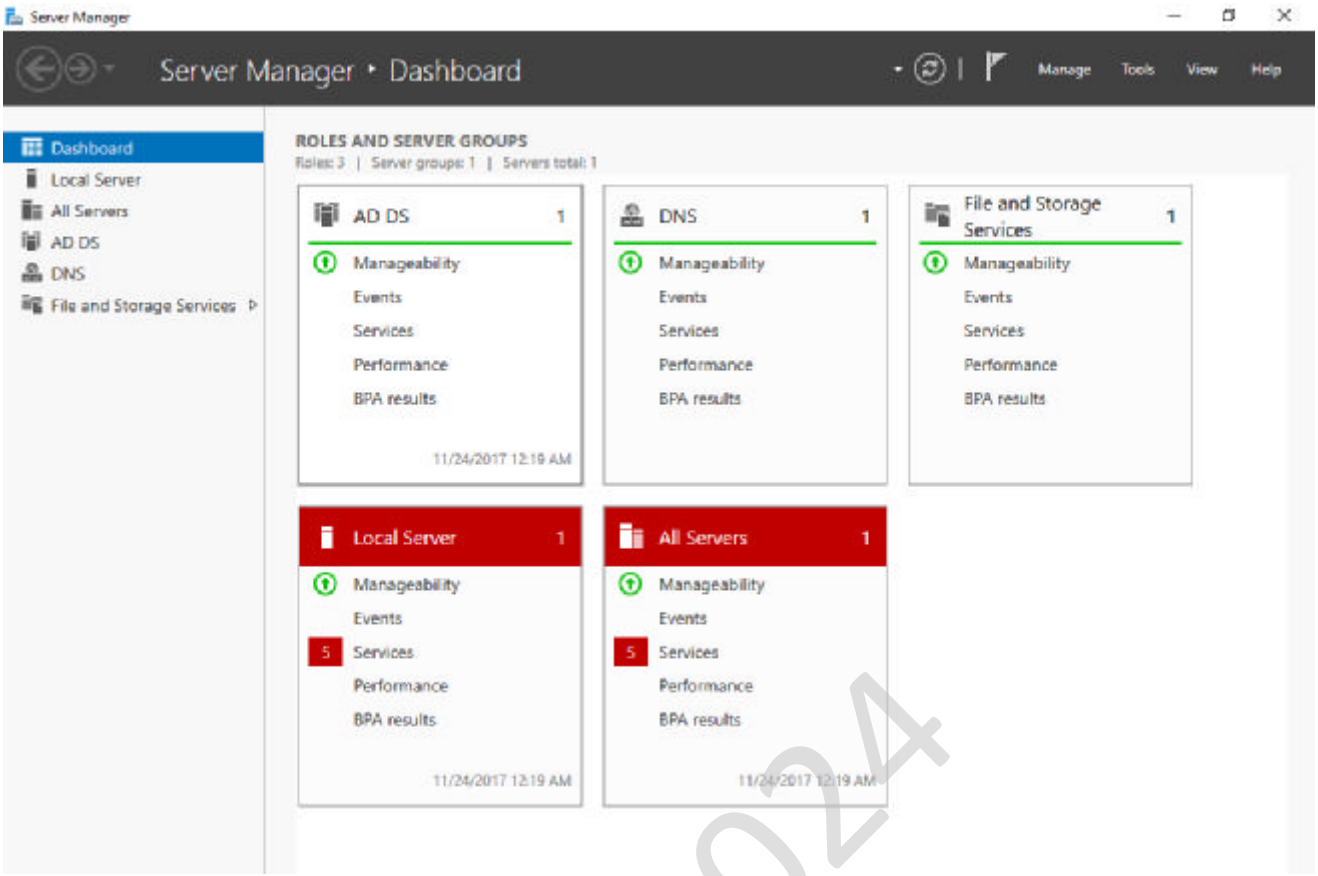

Рисунок 2.2 - Інтерфейс роботи програмного забезпечення Microsoft NPS

Microsoft NPS - це серверна технологія для управління політиками доступу та автентифікації, яка використовується для контролю доступу до ресурсів у мережі. NPS підтримує протоколи RADIUS і TACACS+ та може працювати як сервер аутентифікації для різних додатків.

Переваги:

1) Інтеграція з Windows Server: Легко інтегрується з іншими сервісами Microsoft, такими як Active Directory.

2) Гнучкість політик: Підтримує налаштування політик доступу залежно від ролей та характеристик користувачів.

3) безкоштовність: NPS входить до складу Windows Server, що робить його безкоштовним для користувачів Windows.

Недоліки:

1) Обмежена платформа: Переважно працює в екосистемі Windows, що може бути недоліком для організацій, що використовують різноманітні операційні системи.

2) Складність налаштування: Конфігурація може бути складною для користувачів, які не мають досвіду роботи з серверами Windows.

Також важливим аспектом є використання 802.1X, що дозволяє реалізувати контроль доступу на основі аутентифікації через мережеві пристрої. 802.1X є важливим стандартом, розробленим IEEE (Institute of Electrical and Electronics Engineers), який забезпечує контроль доступу до мережевих ресурсів на основі аутентифікації. Цей протокол активно використовується в сучасних мережах Ethernet та Wi-Fi, забезпечуючи підвищений рівень безпеки шляхом перевірки ідентичності користувачів та пристроїв перед наданням їм доступу до локальної мережі.

Принцип роботи 802.1X базується на моделі клієнт-сервер, де клієнт (пристрій, який намагається підключитися до мережі) повинен пройти аутентифікацію[8]. Весь процес аутентифікації реалізується через RADIUS сервер, який перевіряє облікові дані користувача. При цьому мережевий пристрій, такий як комутатор або точка доступу, виступає в ролі контролера порту, що регулює доступ до мережі на основі результатів аутентифікації.

Стандарт 802.1X надає можливість не лише аутентифікації користувачів, але й розширює можливості безпеки мережі шляхом впровадження політик доступу, які можуть бути адаптовані в залежності від конкретних вимог організації. Це включає в себе контроль доступу до певних ресурсів на основі ролей користувачів, типу пристрою та інших параметрів.

В умовах, коли безпека мереж є критично важливою, 802.1X дозволяє організаціям зменшити ризик несанкціонованого доступу, захищаючи чутливу інформацію та ресурси. У поєднанні з іншими рішеннями, такими як FreeRADIUS, цей стандарт стає невід'ємною частиною стратегій безпеки корпоративних мереж, що забезпечує надійний захист від загроз, пов'язаних із доступом до мережі.

В останні роки з'являються нові технології, такі як OAuth і OpenID Connect, які використовуються для аутентифікації та авторизації в веб-додатках.

OAuth і OpenID Connect - це сучасні протоколи, які забезпечують безпечну аутентифікацію та авторизацію в веб-додатках. Вони дозволяють користувачам використовувати один обліковий запис для доступу до різних сервісів, спрощуючи процес входу в різноманітні додатки, при цьому забезпечуючи високий рівень безпеки[8].

OAuth - це протокол авторизації, який дозволяє стороннім додаткам отримувати доступ до ресурсів користувача без необхідності розкривати його облікові дані. Наприклад, ви можете надати доступ до вашого профілю у Facebook сторонньому додатку, не вводячи пароль. Це забезпечує зручність і безпеку, оскільки користувач може контролювати, які дані будуть доступні для інших додатків.

OpenID Connect розширює можливості OAuth, надаючи додаткові функції для аутентифікації. Це означає, що окрім авторизації, OpenID Connect дозволяє визначити особу користувача, забезпечуючи інтеграцію з існуючими системами управління ідентичністю. Наприклад, якщо ви входите в веб-додаток за допомогою свого облікового запису Google, OpenID Connect дозволяє додатку отримати вашу інформацію, таку як ім'я та електронна адреса, без необхідності створення нового облікового запису.

Обидві технології значно спростили процес аутентифікації та авторизації, підвищивши рівень безпеки та зручності для користувачів. Використовуючи ці протоколи, організації можуть зосередитися на забезпеченні безпеки своїх ресурсів, одночасно спрощуючи доступ для кінцевих користувачів.

З розвитком IoT (Інтернет речей) зростає потреба у нових системах аутентифікації, які можуть працювати з великою кількістю пристроїв. Рішення, що використовують JSON Web Tokens (JWT), стають популярними для аутентифікації в мобільних та веб-додатках, оскільки вони забезпечують легкий обмін даними між клієнтами та серверами.

Таким чином, ринок рішень для аутентифікації користувачів в інтернет-мережах є динамічним і постійно розвивається. FreeRADIUS залишається одним з найбільш надійних і гнучких інструментів у цій сфері, надаючи можливості для інтеграції з різними протоколами та технологіями.

Розробка системи аутентифікації на основі FreeRADIUS дозволить забезпечити безпеку, масштабованість і ефективність управління доступом до мережевих ресурсів, що є критично важливими в умовах швидко змінюваного технологічного середовища.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

У рамках розробки системи автентифікації користувачів на базі FreeRADIUS, одним із важливих рішень є вибір мови програмування. У цьому проекті акцент буде зроблено на Perl та Python. Обидві мови мають свої особливості, які впливають на реалізацію функцій автентифікації.

Perl є потужним інструментом для обробки тексту та роботи з даними. Завдяки своїй простоті у написанні скриптів та підтримці регулярних виразів, Perl часто використовується для автоматизації різних завдань, включаючи управління користувачами.

Основні можливості та властивості Perl

1) обробка тексту: Perl має потужні інструменти для роботи з текстовими даними, включаючи регулярні вирази, що дозволяють легко аналізувати та маніпулювати текстом[3];

2) кросплатформеність: Perl доступний на різних операційних системах, включаючи Windows, macOS, Linux і UNIX, що робить його гнучким для розробників[3];

3) велика кількість бібліотек: Завдяки CPAN (Comprehensive Perl Archive Network) Perl має величезну кількість готових модулів, що значно полегшує розробку[3];

4) об'єктно-орієнтоване програмування: Perl підтримує об'єктно-орієнтовані парадигми, що дозволяє організовувати код у зручні для роботи класи та об'єкти;

5) синтаксис: Синтаксис Perl часто описується як "досить гнучкий", що дозволяє програмістам вирішувати завдання різними способами, хоча це може ускладнювати читабельність коду[3];

6) використання Perl в асинхронному режимі з FreeRADIUS.

FreeRADIUS є одним з найпопулярніших RADIUS-серверів у світі, і Perl може бути використаний для його налаштування та інтеграції, зокрема в асинхронних сценаріях:

1) обробка запитів - Perl може бути використаний для написання обробників запитів, які виконують автентифікацію, авторизацію та облік (AAA). Це забезпечує гнучкість у налаштуванні логіки автентифікації;

2) асинхронні модулі - за допомогою модулів, таких як AnyEvent, програмісти можуть реалізувати асинхронну обробку запитів у Perl, що дозволяє обробляти кілька запитів одночасно, зменшуючи затримки та покращуючи продуктивність;

3) інтеграція з базами даних - Perl легко взаємодіє з різними базами даних, такими як MySQL, PostgreSQL, і може бути використаний для зберігання або отримання даних про користувачів, що важливо для роботи з FreeRADIUS;

4) автоматизація процесів - Perl може автоматизувати задачі, пов'язані з управлінням користувачами, зміною паролів, веденням журналів та звітами, що робить його корисним для адміністраторів мереж.

Perl активно використовується в різних сферах:

- веб-розробка: створення динамічних веб-додатків (на основі модулів, таких як Dancer або Mojolicious);

- системне адміністрування: автоматизація рутинних завдань та управління системами;

- наукові дослідження: обробка великих обсягів даних та їх аналіз.

Таким чином, Perl є потужним інструментом для розробників, особливо в контексті налаштування та інтеграції систем, таких як FreeRADIUS, завдяки своїм можливостям обробки тексту, асинхронності та підтримці різних модулів.

Python, з іншого боку, славиться своїм простим синтаксисом і великою кількістю бібліотек, що дозволяє швидко реалізовувати складні рішення. Його популярність зростає, особливо у сфері веб-розробки та роботи з даними, що робить його підходящим вибором для інтеграції з системою FreeRADIUS.

Основні можливості та властивості Python:

1) читабельність і зрозумілість коду: Python використовує чіткий та зрозумілий синтаксис, що дозволяє програмістам зосередитися на розв’язанні проблем, а не на нюансах мови;

2) велика бібліотека стандартних модулів: Python постачається з широким набором стандартних бібліотек, які спрощують виконання багатьох завдань, від обробки тексту до роботи з мережею;

3) підтримка різних парадигм програмування: Python підтримує об'єктно-орієнтоване, процедурне та функціональне програмування, що робить його універсальним для різних типів проектів;

4) кросплатформеність: Python доступний на різних платформах, що дозволяє програмістам розробляти кросплатформені рішення без суттєвих змін у коді;

5) активна спільнота та підтримка: Python має велику і активну спільноту, що забезпечує численні ресурси, такі як документація, онлайн-курси, форуми та бібліотеки;

6) використання Python в асинхронному режимі з FreeRADIUS.

FreeRADIUS є популярним сервером аутентифікації, і Python може бути використаний для його налаштування та інтеграції, зокрема для реалізації асинхронної обробки запитів.

FreeRADIUS складається з декількох модулів, кожен з яких виконує конкретну роль в обробці запитів автентифікації. Основні блоки:

1. **freeraidusd** - цей блок є основою всієї системи. Він відповідає за:

- обробку запитів: отримує запити від клієнтів, аналізує їх та передає у відповідні модулі;

- роботу з конфігураційними файлами: читає налаштування з файлів, таких як radiusd.conf і clients.conf;

- логування: записує інформацію про успішну та неуспішну автентифікацію, помилки, відхилені запити.

2. auth (автентифікація) - цей блок перевіряє, чи є користувач дійсним і чи відповідає його пароль/ключ необхідним вимогам.

- підтримка протоколів: PAP, CHAP, MS-CHAP, EAP;

- робота з базами даних: може використовувати зовнішні джерела даних (LDAP, SQL, JSON);

- налаштування правил: перевірка політик доступу (наприклад, перевірка MAC-адрес або часу доступу);

PAP (Password Authentication Protocol) – це простий протокол автентифікації, який передає ім'я користувача та пароль у відкритому вигляді. Через відсутність шифрування PAP вважається небезпечним, тому його зазвичай застосовують у середовищах, де передача даних захищена іншими способами, наприклад, через VPN або TLS. Попри низький рівень безпеки, протокол залишається популярним завдяки простоті налаштування та сумісності із багатьма пристроями.

CHAP (Challenge-Handshake Authentication Protocol) використовує механізм взаємодії, що базується на запиті та відповіді. Клієнт отримує випадковий запит (challenge) від сервера, обчислює на його основі хеш пароля та

надсилає результат назад. Сервер, маючи хеш, перевіряє автентичність клієнта.

Цей протокол є безпечнішим за PAP, оскільки пароль не передається відкритим текстом. Однак CHAP вразливий до атаки, якщо зловмисник отримує доступ до хешу, а також вимагає збереження пароля у незашифрованому вигляді на сервері.

MS-CHAP (Microsoft Challenge-Handshake Authentication Protocol) – це розширення CHAP, розроблене Microsoft. Воно забезпечує вищий рівень безпеки завдяки використанню протоколу NTLM для хешування пароля. MS-CHAP часто застосовується у VPN-сервісах і корпоративних мережах.

Однак його друга версія, MS-CHAPv2, стала стандартом через усунення вразливостей оригінальної версії. MS-CHAPv2 підтримує двосторонню автентифікацію, що дозволяє перевіряти автентичність як клієнта, так і сервера.

EAP (Extensible Authentication Protocol) є універсальним фреймворком для автентифікації, який підтримує різні методи, включаючи сертифікати, паролі, токени і біометричні дані. EAP не передає паролі безпосередньо, натомість використовує захищені канали для їх передачі. Завдяки гнучкості, EAP застосовується у бездротових мережах (наприклад, WPA/WPA2-Enterprise) та мережах, що вимагають високого рівня безпеки. Серед його популярних методів – EAP-TLS (на основі сертифікатів) та PEAP (Protected EAP), який додає захист через тунелювання.

3. post-auth (після автентифікації) - цей модуль виконує дії після успішної автентифікації:

- логування: запис результатів у базу даних або файли журналу;

- динамічне управління: додавання правил (наприклад, обмеження на час сесії);

- обробка відповідей: відправка клієнту додаткових атрибутів, таких як VLAN ID чи QoS параметри.

4. authorize (авторизація) - Цей модуль перевіряє, чи має користувач дозвіл на доступ до певних ресурсів або послуг:

- правила доступу: визначення, які ресурси або сервіси доступні для користувача;

- робота з профілями: прив'язка до групових або індивідуальних налаштувань (наприклад, тарифні плани);

- використання зовнішніх джерел: інтеграція з LDAP, AD чи іншими системами авторизації.

Вибір СКБД. СКБД - програми засновані на базах даних і варіанти використання даних (вставка, видалення, редагування, вибірки).

Система є досить безпечною та відповідає всім критеріям цілісності даних.

Основні функції реляційної СКБД:

- безпосередньо управління даними в пам'яті;

- керувати операціями буфера;

- керувати транзакціями;

- підтримка мови SQL.

Класифікація баз даних:

- ієрархічні;

- мережеві;

- реляційні;

Проаналізувавши вищезазначене, було обрано дві основні СКБД - MySQL та PostgreSQL, які підтримують всі основні функції систем керування баз даних. MySQL - реляційна, PostgreSQL - об'єктно–реляційна.

Розглянемо їх можливості. Обидві використовують мову SQL в якості мови запитів до баз даних/таблиць.

SQL - декларативна мова запитів, що застосовується для створення, модифікації та управління даними в реляційній БД, керованої відповідною системою управління базами даних.

Є, перш за все, інформаційно–логічною мовою, яка призначена для опису, зміни і вилучення даних, що зберігаються в базах даних.

Базові функції мови:

- створення в базі даних нової таблиці;

- зміна структур таблиць;

- додавання в таблицю нових записів;

- вибірка записів з однієї або декількох таблиць (відповідно до заданого умовою);

- зміна записів;

- видалення записів.

MySQL - є системою управління реляційними базами даних з відкритим вихідним кодом. MySQL базується на моделі клієнт-сервер. Основним моментом використання MySQL є можливості обробки даних.

При виконанні запиту MySQL завантажує всю відповідь сервера в пам'ять клієнта, при великих обсягах даних це може бути не зовсім зручно. У більшості випадків для організації роботи з базою даних в MySQL використовується таблиця InnoDB, ця таблиця представляє з себе B-дерево з індексами.

Індекси дозволяють швидко отримати дані з диска, це призводить до зменшення використання обчислювальної потужності. Але сканування дерева вимагає знаходження двох індексів, а це вже повільно.

PostgreSQL - вільна об'єктно-реляційна система управління базами даних Основною характеристикою PostgreSQL є використання індексів. У PostgreSQL є підтримка індексів таких як: хеш, B-дерево. При необхідності можна створювати нові типи індексів.

Індекси в PostgreSQL мають властивості:

- можливий перегляд індексу не тільки в прямому, а й у зворотному порядку;

- можливе створення індексу над декількома стовпцями таблиці, в тому числі над стовпцями різних типів даних;

- індекси можуть бути функціональними, тобто будуватися не на базі набору значень якогось стовпця/стовпців, а на базі набору значень функції від набору значень;

Планувальник запитів може використовувати кілька індексів одночасно для виконання складних запитів. Розглянемо основні моменти використання PostgreSQL.

Можливості обробки PostgreSQL підтримує використання курсорів для переміщення по отриманим даним – отримує тільки покажчик, а вся відповідь зберігається в пам'яті сервера баз даних.

Цей покажчик можна зберігати між сеансами. Тут підтримується побудова індексів відразу для декількох стовпців таблиці. Продуктивність Вся заголовна інформація таблиць PostgreSQL знаходиться в оперативній пам'яті.

Отже, не можна створити таблицю, яка буде не в пам'яті. Записи таблиці упорядковано відповідно по індексу, а тому можна їх дуже швидко витягти. Для більшої зручності прийнято застосовувати кілька індексів до однієї таблиці.

В цілому PostgreSQL працює швидше, за виключенням використання первинних ключів.

Отже, проаналізувавши переваги та недоліки вищезазаначених технологій та посилаючись на завдання даного дипломного проекту, мовою обрано Perl з СКБД MySQL.

2.3 Розгорнута постановка завдання

Сучасний стан систем автентифікації демонструє високу актуальність та ефективність, особливо у використанні таких рішень, як FreeRADIUS. Цей інструмент широко застосовується в сфері інформаційної безпеки для забезпечення надійного управління доступом до мережевих ресурсів.

FreeRADIUS підтримується провідними компаніями у сфері розробки програмного забезпечення, що підтверджує його релевантність у сучасних IT-системах.

У межах проєкту, присвяченого дослідженню та програмній реалізації системи автентифікації користувачів в інтернет-мережах, буде використано FreeRADIUS для створення ефективної системи автентифікації.

Цей проект виконується згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти.

У процесі виконання роботи необхідно вирішити наступні завдання:

1) аналіз та вивчення теоретичних основ автентифікації в інтернет-мережах;

2) проектування архітектури системи автентифікації з використанням FreeRADIUS;

3) розробка структури бази даних для зберігання облікових записів користувачів;

4) налаштування та конфігурування FreeRADIUS для роботи з базою даних;

5) інтеграція FreeRADIUS з клієнтськими та серверними компонентами системи;

6) тестування системи автентифікації на реальних сценаріях використання;

7) забезпечення високого рівня безпеки шляхом впровадження сучасних криптографічних алгоритмів і протоколів.

Результатом виконання роботи стане готова до використання система автентифікації користувачів на основі FreeRADIUS, яка відповідає сучасним вимогам безпеки та функціональності.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У сучасному світі, де інтернет став невід'ємною частиною повсякденного життя, важливо забезпечити безпечний доступ до мережевих ресурсів. Система автентифікації на основі FreeRADIUS є ефективним рішенням для управління доступом користувачів. Основними компонентами цієї системи є:

- Серверна частина.
- Сховище.
- Система білінгу.
- Модуль Perl для FreeRADIUS.
- Модуль datetime.
- Видалений доступ.

Серверна частина - основний компонент, який виконує функції обробки запитів на автентифікацію, авторизацію та облік (AAA). FreeRADIUS є гнучким і масштабованим сервером, який підтримує різноманітні протоколи автентифікації.

Сховище - база даних, де зберігаються дані про користувачів, їх паролі, права доступу та інші параметри.

Система біллінгу - компонент, що відповідає за облік та управління фінансовими аспектами доступу до послуг. Система білінгу може автоматично відстежувати використання ресурсів і генерувати рахунки для користувачів.

Модуль Perl для FreeRADIUS - використовується для розширення функціональності сервера та налаштування логіки обробки запитів. Perl дозволяє інтегрувати зовнішні системи та додаткові модулі.

Модуль datetime - цей модуль забезпечує управління часом і датою в запитах та відповідях, що є важливим для моніторингу активності користувачів.

Видаленний доступ - механізм, який дозволяє користувачам отримувати доступ до системи з віддалених локацій.

Система працює на базі основної бібліотеки freeradius, за допомогою якої виконується реалізація суттєвих рішень в автентифікації. Для початку в основному файлі, а саме clients.conf, описується загальний список всіх доступних пристроїв котрі мають можливість підключення до системи.

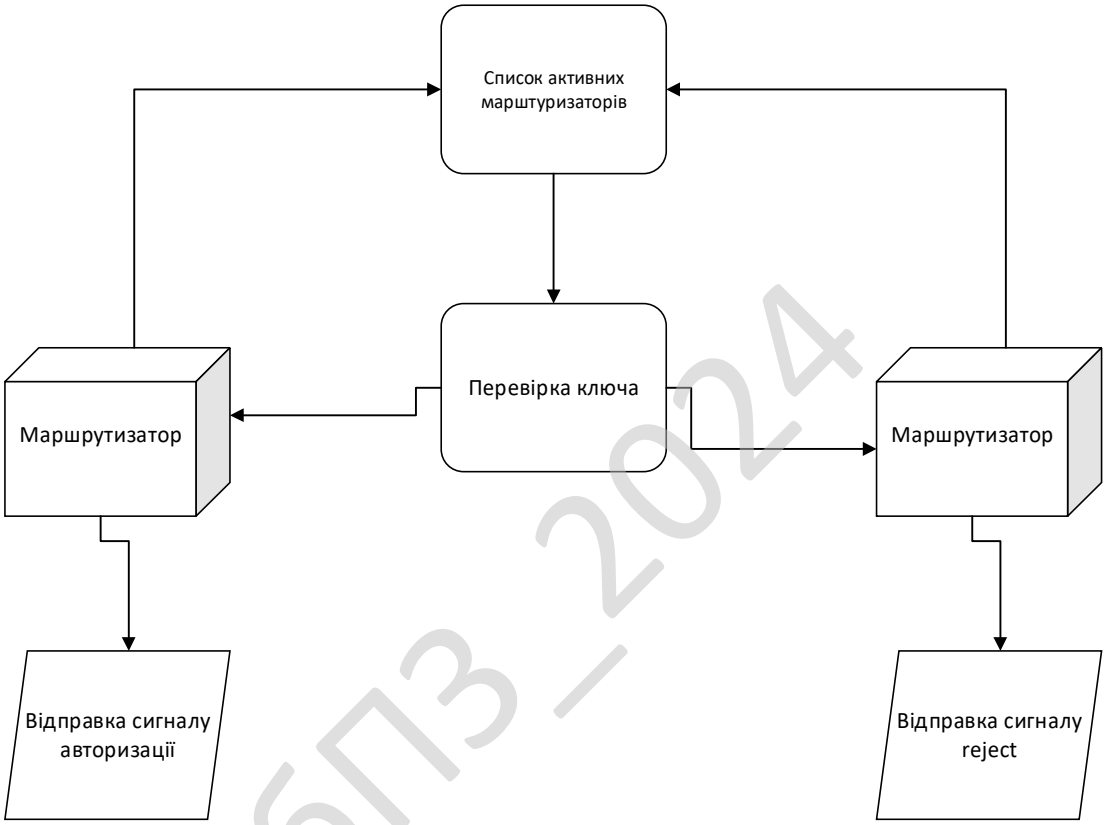

Рисунок 3.1 - Процес авторизації клієнтів через перевірку списку активних маршрутизаторів

На рисунку 3.1 зображений процес авторизації клієнтів, а саме тих пристроїв що можуть відправляти дані на сервер. Кожен пристрій має свою ip адресу та секретний ключ що вказується у його конфігураціях, ці дані вказується у самому списку.

Параметри формування списку активних клієнтів:

- ip адресса;

- назва клієнта;

- секретний ключ;

- тип марштрутизатора (його операційна система);

- максимальна кількість одночасних запитів.

Після загальної перевірки клієнту, запит переходіть до блоку в котрому описана схема проходження всіх сигналів. А саме:

- authorize;

- authrenticate;

- preacct;

- accounting;

- session;

- post-auth.

Кожен блок відповідає за свій процесс:

@authorize – система перевіряє логин та пароль користувача, за методом обраним у конфігураційному файлі, це може бути PAP або CHAP.

PAP – зберігає пароль та логин у чистому вигляді, без шифрування. Потім ці дані порівнюється із інформацією що міститься у базі даних.

CHAP – працює подібно PAP, але отриманий пароль та логин шифрується за допомогою певного протоколу. Для порівяння інформації, пароль що зберігався у базі даних, також проходить шифрування, після чого 2 шифри порівнується між собою по об’єму самого шифру.

Якщо користувач успішно пройшов перевірку паролю, перевіряється наявність вже активного підключення, якщо воно існує, блок завершується. У випадку якщо підключення не знайдено, створюється підключення до бази даних та перевіряється баланс користувача та його тарифний план. Після успішного прорахування суми, із бази даних отримується список доступних ip адрес, та передається у параметрах для наступного блоку, також в список параметрів входить тарифний пакет, швидкість підключення, та видаленний сервер котрий буде виконувати роль DNS серверу. Блок завершується, та до маршрутизатора передається сигнал про можливість виконання блоку @accounting.

@authenticate – може бути викликаний тільки після успішного проходження блоку @authorize. Із отриманих даних, перевіряється ключ $RAD_REQUEST{'Framed-Protocol'} – який містить метод за яким користувач бажає виконати підключення, це може бути як IpoE (dhcp) так і (PPPoE).

@preacct – блок який викликається перед @authorize – для додаткового функціоналу. У данному проєкті – не використовується.

@accounting – фінальний блок у даному проєкті. Блок викликається тільки після проходження блоку @authenticate. На цьому етапі створюється підключення за вже сформованими параметрами, та зберігається у базі даних як активне підключеня за логином користувача.

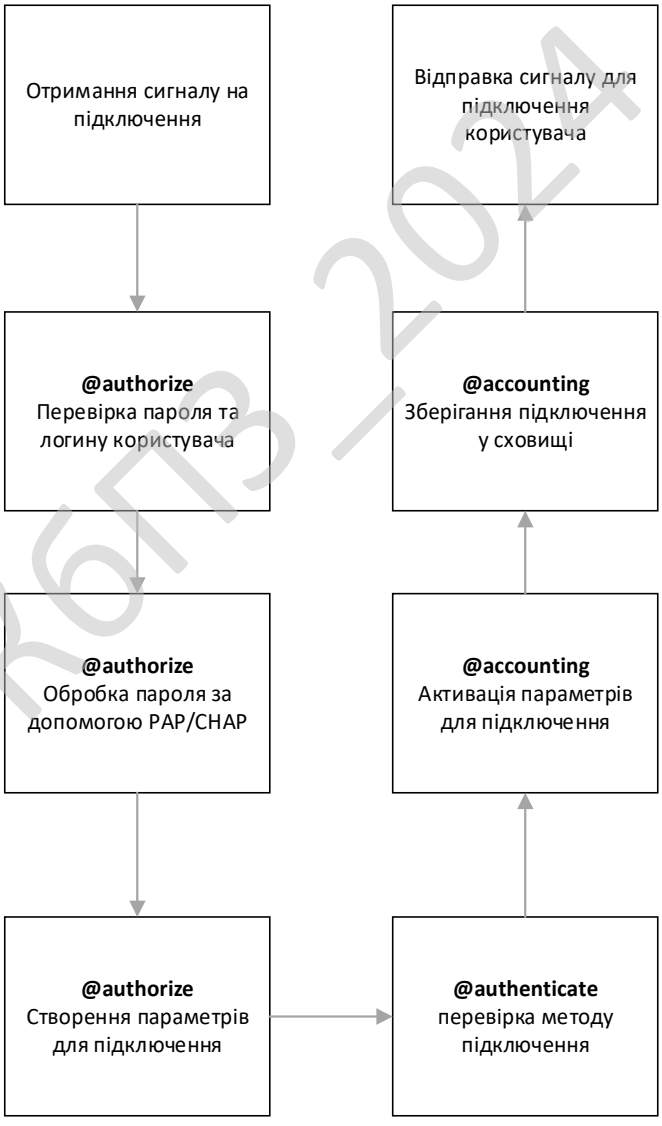

Рисунок 3.2 – Схема проходження всіх блоків функціонування системи

Для роботи з активними сеансами маршрутизатора використовується бібліотека Redis. Redis є високопродуктивним ключ-значення сховищем, яке ідеально підходить для роботи з даними в режимі реального часу. У контексті взаємодії з активними сеансами маршрутизатора Redis використовується для:

1. **Зберігання інформації про активні сесії.** У Redis зберігаються дані про поточні сеанси, такі як:

- Ідентифікатор сесії (Session ID).

- Ідентифікатор клієнта (наприклад, IP або MAC-адреса).

- Таймстамп початку сеансу.

- Параметри авторизації (ключ доступу, права тощо).

Ці дані зберігаються у форматі ключ-значення, де ключем є унікальний ідентифікатор сесії, а значенням – JSON-об'єкт із деталями сесії.

2. **Оперативна перевірка активності клієнтів**. При кожному запиті маршрутизатор перевіряє Redis на наявність активного сеансу для клієнта.

Якщо запис існує, то:

- Виконується перевірка ключа доступу.

- Оновлюється інформація про останню активність.

У разі відсутності запису клієнт отримує сигнал reject.

3. **Управління сесіями.** Redis дозволяє керувати життєвим циклом сесій через TTL (Time-to-Live). Після завершення терміну дії сесії Redis автоматично видаляє запис, що спрощує очистку застарілих даних.

4. **Масштабованість та продуктивність.** Завдяки своїй високій швидкодії Redis забезпечує обробку великої кількості запитів у реальному часі, що робить його ідеальним вибором для масштабованих систем.

5. **Простота інтеграції.** Бібліотека для роботи з Redis легко інтегрується з будь-якими сучасними мовами програмування, включаючи Python. Використовуються основні команди, такі як:

- SET для збереження нової сесії.

- GET для отримання даних про сесію.

- DEL для видалення сесії.

- EXPIRE для встановлення часу життя запису.

Таким чином, використання Redis дозволяє створити ефективну, масштабовану та надійну систему взаємодії з активними сеансами маршрутизатора.

3.2 Розробка структурної схеми

Полягаючи на всі функціональні та структурні взаємозв'язки побудуємо структурну схему нашої системи

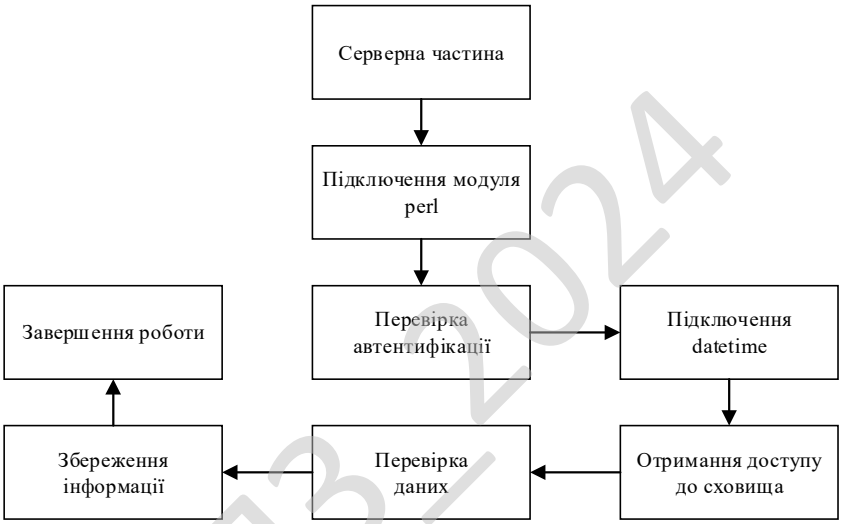

Рисунок 3.3 – Структурна схема система

На рисунку 3.1 наведена структурна схема системи. Вона побудована покладаючись на взаємозв'язок усіх процесів додатку. За допомогою розробки функціональних частин можна окреслити зв’язки між ними.

Опис структурної схеми системи:

1. Серверна частина - основний компонент, який відповідає за обробку всіх запитів. Серверна частина забезпечує централізовану взаємодію з модулями та перевірку даних, необхідних для роботи системи.

2. Підключення модуля Perl - на цьому етапі серверна частина ініціалізує модуль Perl, який використовується для реалізації бізнес-логіки. Завантаження модуля дозволяє використовувати специфічні функції та методи для подальшої обробки даних.

3. Перевірка автентифікації - система перевіряє, чи відповідають надані клієнтом дані встановленим критеріям доступу. Це включає перевірку логіну, пароля або іншого способу ідентифікації залежно від налаштованого механізму.

4. Підключення datetime - цей модуль забезпечує отримання поточної дати й часу, що може бути використано для обчислення тривалості сеансу, створення міток часу для записів або верифікації доступу.

5. Отримання доступу до сховища - після успішної автентифікації виконується запит до сховища даних, де зберігаються необхідні для роботи системи відомості. Це може включати конфігураційні параметри або користувацьку інформацію.

6. Перевірка даних - верифікуються отримані дані із запиту. Перевірка включає порівняння вхідної інформації зі збереженими записами в базі, перевірку формату даних та їхньої актуальності.

7. Збереження інформації - результати перевірки та обробки записуються до бази даних або лог-файлів. Цей крок важливий для забезпечення прозорості системи та можливості відновлення даних у разі потреби.

8. Завершення роботи - фінальний етап, на якому система закриває з'єднання, звільняє ресурси та завершує обробку поточного запиту. Після цього система готова до обробки наступного запиту.

3.3 Розробка функціональної схеми

Функціональна схема відображає роботу усіх компонентів програмного забезпечення між собою, опис інформаційних складових що використовуються.

Усі взаємодії між процесами та сервісами можуть бути виконані через автоматично, але потрібно пройти автентифікацію.

Процес автентифікації виконує функцію спрямовану на забезпечення безпеки інформації. Автентифікації користувача включає в себе отримання логіну та паролю.

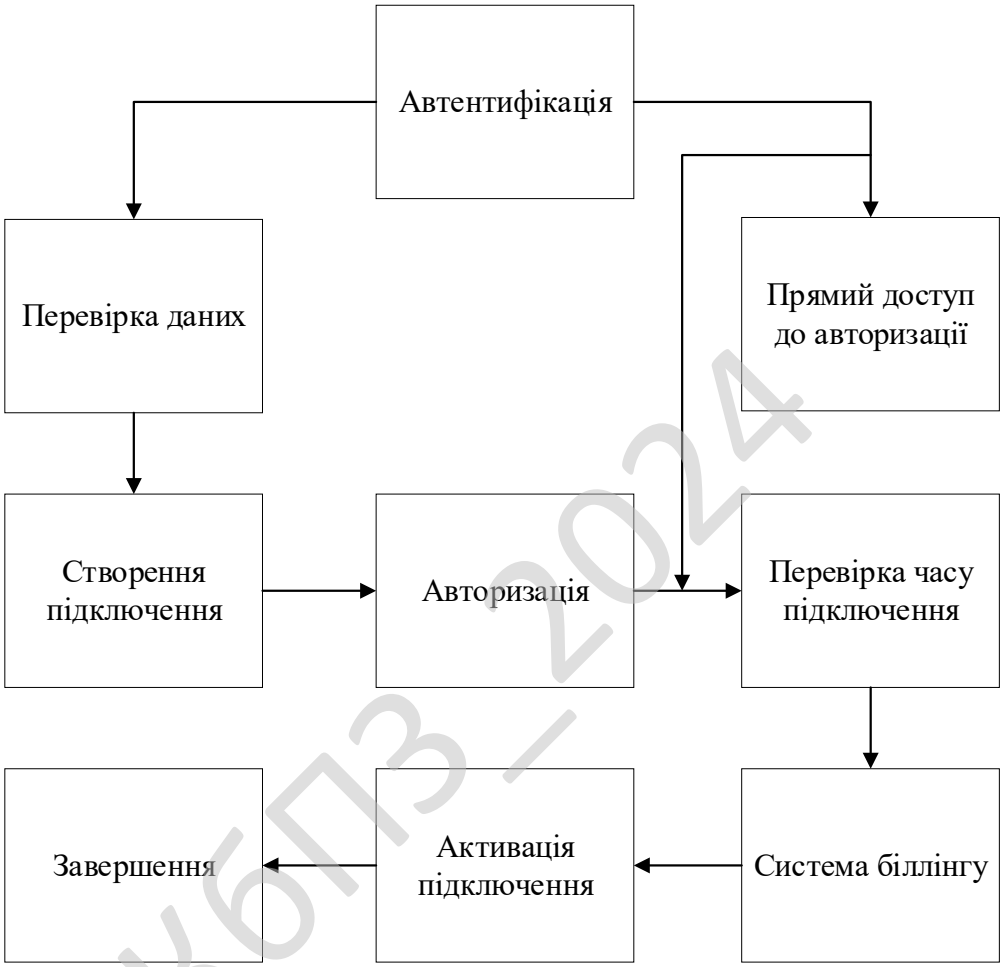

Рисунок 3.4 – Функціональна схема системи

Функціональна схема система описує функціональні можливості програмного продукту. Вона дозволяє відобразити зв’язки між компонентами та принцип їх роботи.

Головна робота програми полягає у авторизації та біллінгу клієнта, із отриманих даних система обирає тільки ті параметри що були вкладені у конфігурацію. Якщо клієнт проходив автентифікації раніше, цей блок може не відп-

рацьовувати, а відразу проходити авторизації, що прискорює загальну роботу усієї системи.

Схема виконання роботи функціонування системи у прикладі.

Користувач починає процес входу до системи. На цьому етапі перевіряються його облікові дані, такі як ім'я користувача та пароль. Відбувається верифікація введених даних (ідентифікація користувача у базі даних або перевірка сертифікатів). Якщо дані валідні, система встановлює з'єднання для подальшої роботи користувача. У деяких випадках користувач може одразу перейти до етапу авторизації без проміжної перевірки.

Визначається рівень доступу користувача до системних ресурсів, виходячи з його ролі чи політики безпеки. Система перевіряє дозволений час доступу користувача, щоб уникнути перевищення лімітів.

Інформація про сесію передається до системи білінгу для обліку використаних ресурсів або тарифікації. Після успішного проходження всіх перевірок з'єднання активується для користувача. Після завершення роботи користувача підключення закривається, і система завершує сесію.

3.4 Розробка діаграми процесів

Для повного розуміння функціоналу та взаємодії класів та функцій між собою, використовуються діаграми процесів, що дає змогу розробнику поліпшити схему початкової розробки.

Діаграма потоків даних – модель проектування, графічне представлення «потоків» даних в інформаційній системі. Діаграма потоків даних також може використовуватись для візуалізації процесів обробки даних.

Спершу ми перевіряємо отримання запиту автентифікації, далі перевіряємо отриманні данні. Блок авторизації є найголовнішим, потрібно створити підключення клієнту, якщо поточний та отриманий час відповідає дійсності, а та-

кож нам потрібно зберегти інформацію у СБКД. Всі процеси взаємопов'язані, кожен блок може бути оброблений поступово або в асинхронному режимі.

Після проходження блоку авторизації, також потрібно пройти блок автентифікації та аккаунтингу. Загальним фактором перевірки усіх блоків, є успішне завершення попереднього, тобто у разі невдачного завершення одного із верхніх блоків, весь процесс завершується, без змоги повторного проходження.

У випадку неуспішного завершення блоків, маршрутизатор запам'ятовує відправлений сигнал, та входить у режим очікування на 1 хвилину, після чого знову відправляє запит для підключення. Дане очікування створено, щоб запобігти перебільшенню потоку виконання.

Таким чином якщо всі блоки були пройдені успішно, користувачу створюється підключення. У цьому підключенні я також таймер, часом зазначеним у файлі конфігурації, після завершення дії таймера, маршрутизатор відправляє сигнал для отримання часу підключення клієнта, що у свою чергу продовжує вже існуючий зв'язок

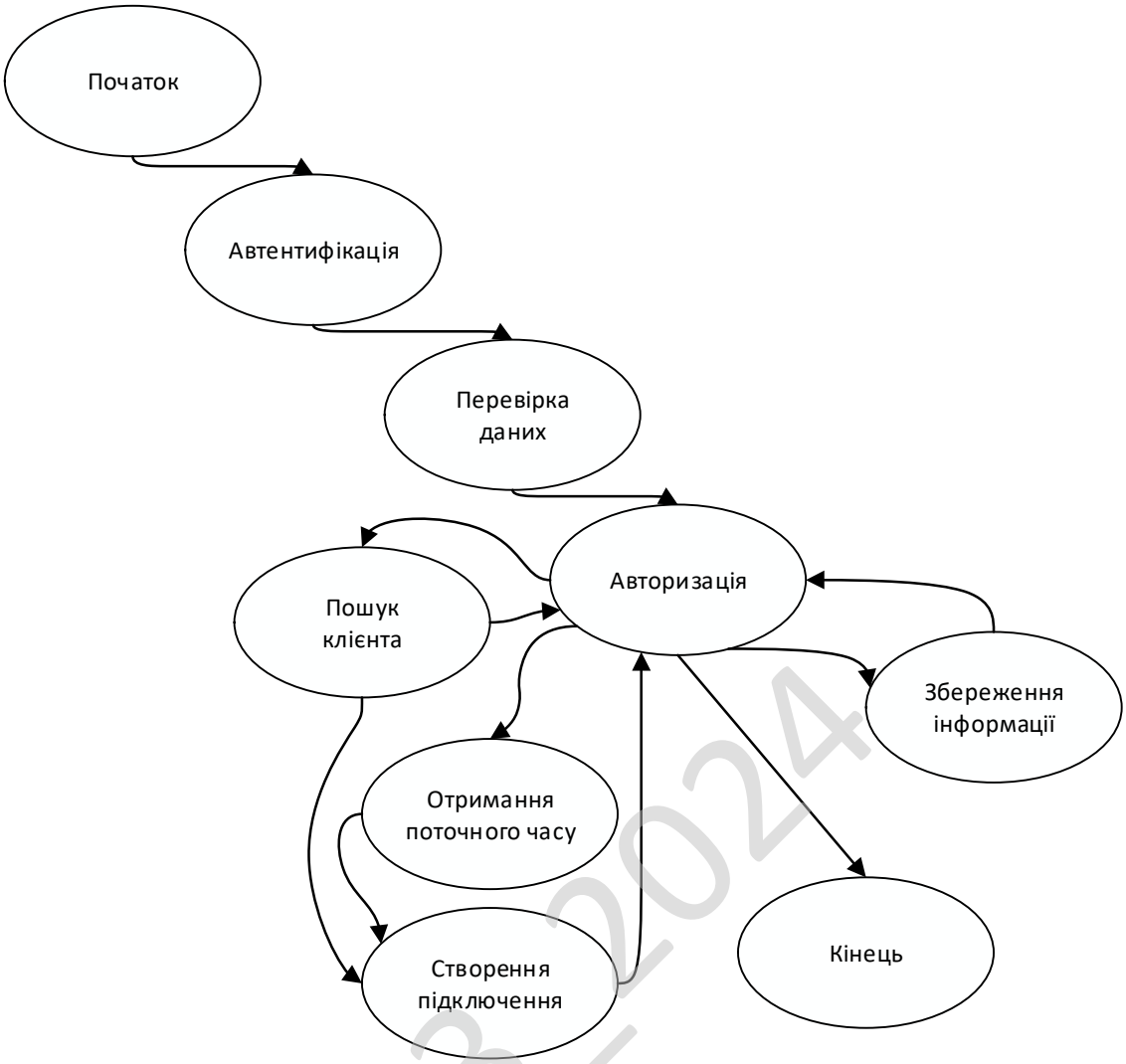

Рисунок 3.5 - Діаграма процесів системи

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Виходячи з розроблених функціональної та принципової електричних схем будуємо алгоритми, за якими повинен працювати система. З поставленої задачі до роботи ми маємо систему управління автентифікації та авторизації користувачів.

Розглянемо загальний алгоритм.

На початку роботи система отримує запит від обладнання, яке відповідає за обробку підключення клієнта. Цей запит містить важливу інформацію про клієнта, включаючи його ідентифікатори та параметри підключення. Після отримання запиту, система проводить перевірку вхідної інформації, щоб визначити, чи може клієнт миттєво перейти до блоку авторизації без проходження етапу автентифікації.

Можливість такого миттєвого переходу реалізується у випадку, якщо клієнт вже раніше відправляв запит на підключення і на даний момент має активну сесію. Якщо сесія активна, система автоматично направляє запит до блоку авторизації, що значно зменшує затримки при підключенні та підвищує зручність для користувача.

У випадку, якщо клієнт не має активного з'єднання, система виконує пошук даних про клієнта в системі керування базами даних (СКБД). Цей процес передбачає перевірку наявності записів, пов'язаних з ідентифікатором клієнта, та аналіз їх стану. Якщо система знаходить інформацію про клієнта, вона переходить до блоку авторизації.

Блок авторизації є одним із найважливіших компонентів системи, оскільки тут перевіряються всі вхідні параметри, які надходять разом із запитом.

Цей блок також виконує функцію інтеграції з білінговою системою, що відповідає за обробку фінансових транзакцій і управління платежами. Під час авторизації система аналізує, чи є у клієнта достатня кількість ресурсів, а також перевіряє кількість доступних IP-адрес.

Якщо всі перевірки проходять успішно, система створює нове підключення, реєструючи дату та час, які отримуються з підключеного модуля. Ця інформація зберігається у СКБД для подальшого моніторингу і аналізу. Завдяки такій архітектурі система забезпечує високий рівень безпеки та надійності, а також оптимізує процес підключення клієнтів до мережі.

В цілому, процес, описаний вище, підкреслює важливість злагодженої роботи всіх компонентів системи, що забезпечує ефективне обслуговування клієнтів і максимальну безпеку під час обробки їх запитів.

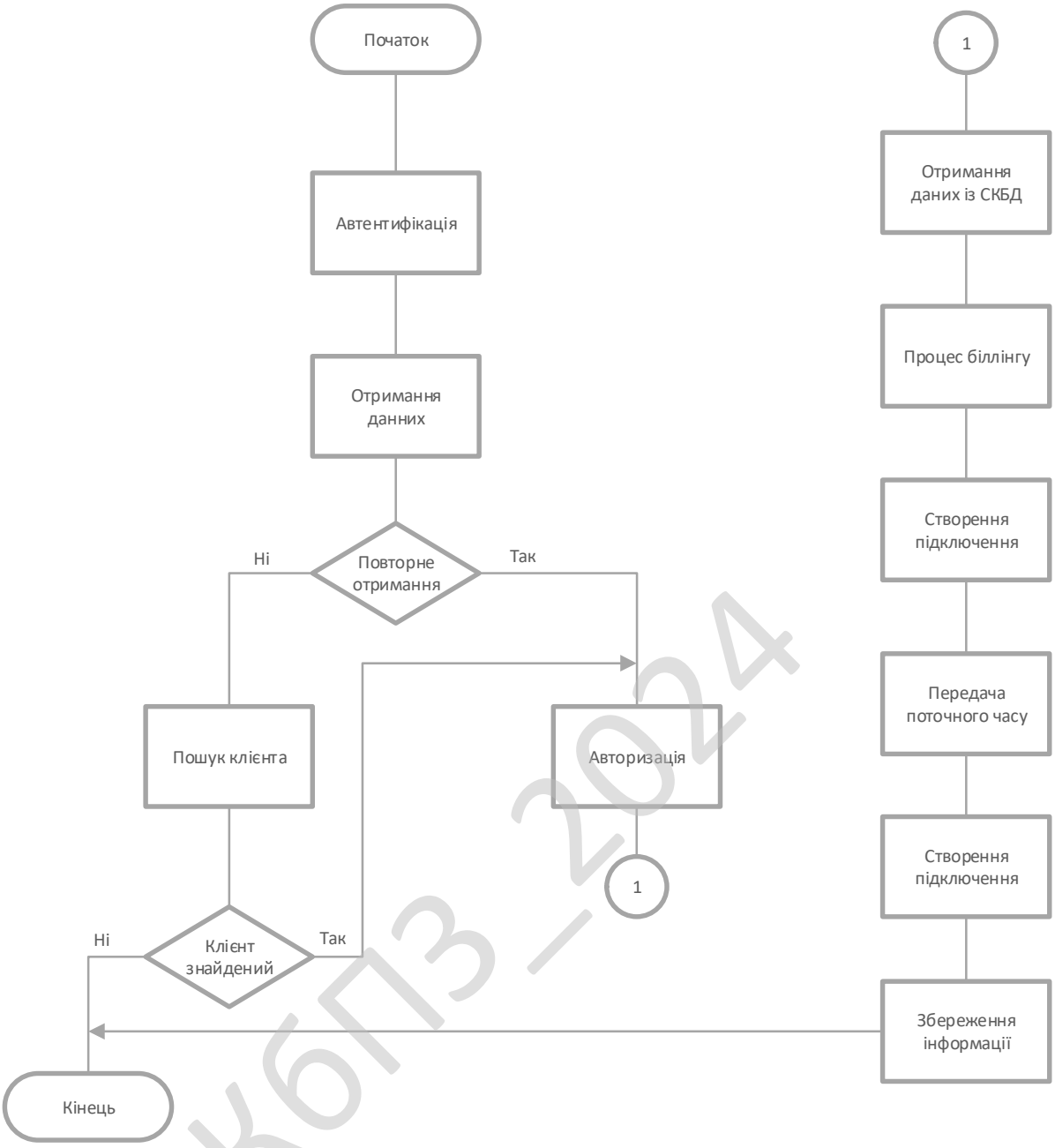

Рисунок 4.1 – Алгоритм авторизації та автентифікації користувачів

Загальний принцип роботи алгоритму полягає у перевірці отриманих даних відправлених клієнтом для авторизації та автентифікації. За допомогою цих даних виконується тарифікація клієнта, та збереження інформації для подальшого використовування. Кожен блок відпрацьовує автоматично, і має переходи до підпрограм.

Розглянемо кожен блок окремо:

1. **Автентифікація.** На цьому етапі здійснюється перевірка облікових даних користувача, що намагається підключитися до системи. Основними діями є:

- Прийом облікових даних клієнта (логін, пароль або інші ідентифікаційні дані).

- Перевірка даних на відповідність стандартам безпеки.

- Верифікація через базу даних або інші зовнішні системи аутентифікації.

- У разі успішної автентифікації користувач переходить до наступного блоку, в іншому випадку - система завершує процес для даного запиту.

2. **Отримання даних.** Цей блок відповідає за отримання інформації, необхідної для продовження роботи:

- Здійснюється запит до джерела даних (наприклад, SQL-база, Redis або файлова система).

- Завантажуються параметри клієнта, зокрема, права доступу, тарифний план, статус сесії.

- Якщо дані не вдалося отримати, то ініціюється блок повторного отримання.

3. **Повторне отримання.** Якщо під час отримання даних виникла помилка або інформація була неповною:

- Система повторює запит до джерела даних.

- Перевіряє альтернативні джерела, якщо такі є (наприклад, репліки бази даних).

- У разі невдачі система переходить до блоку "Пошук клієнта".

4. **Пошук клієнта.** На цьому етапі система намагається знайти клієнта за додатковими параметрами:

- Ідентифікація може здійснюватися за MAC-адресою, IP-адресою або іншими метаданими.

- Якщо клієнт знайдений, процес продовжується до авторизації; якщо ні - запит завершується.

5. **Клієнт знайдений.** Цей блок перевіряє, чи знайдений клієнт у попередніх етапах:

- Якщо клієнт знайдений — система переходить до авторизації.

- У разі відсутності клієнта процес завершується.

6. **Авторизація.** Авторизація визначає права клієнта на виконання певних дій:

- Перевіряються правила доступу клієнта (наприклад, чи має він активний тарифний план).

- Застосовуються політики доступу на основі конфігураційних файлів (наприклад, users.conf у FreeRADIUS).

- Якщо авторизація успішна, запит переходить до наступного етапу.

7. **Отримання даних із СКБД.** Після успішної авторизації виконується запит до СКБД для отримання додаткової інформації:

- Завантаження даних про поточний тариф клієнта, залишок коштів на рахунку тощо.

- Перевірка, чи достатньо ресурсу для встановлення підключення.

8. **Процес білінгу.** Відбувається облік коштів користувача:

- Фіксується поточний баланс клієнта.

- Розраховується час використання або спожитий трафік відповідно до тарифного плану.

- У разі недостатнього балансу з'єднання може бути припинене.

10. **Передача поточного часу**. Система фіксує початок сесії та передає дані про поточний час:

- Використовується для обліку тривалості сесії.

- Передається як параметр для подальшої взаємодії між клієнтом і сервером.

11. **Збереження інформації.** Фінальний етап, на якому зберігається інформація про клієнта та його підключення:

- Фіксуються деталі сесії (час початку, параметри доступу, використані ресурси).

- Інформація зберігається у базі даних або лог-файлах для подальшого аналізу або білінгу.

Цей процес є послідовним і автоматизованим, що забезпечує стабільність роботи та високу швидкість обробки запитів.

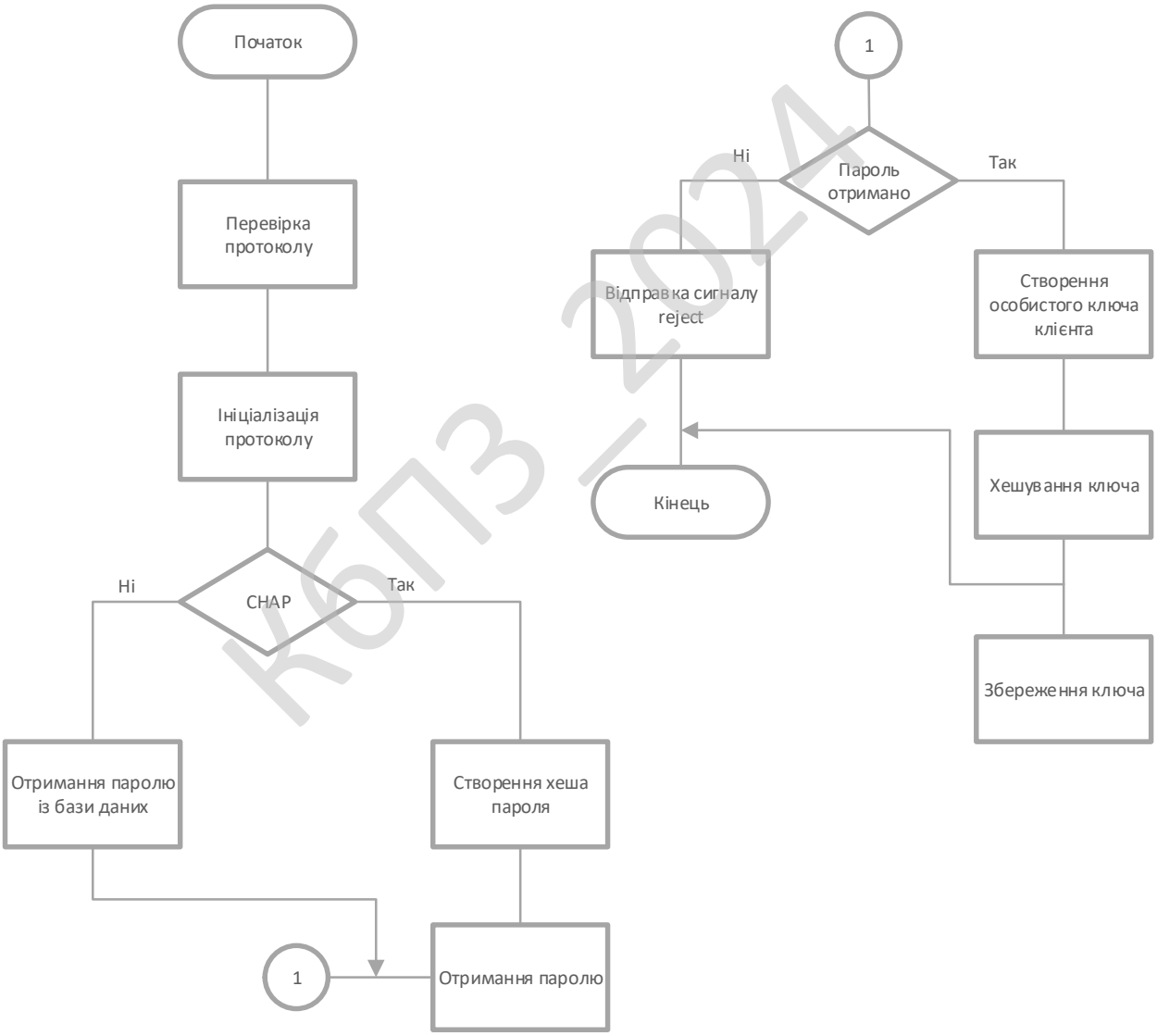

Рисунок 4.2 - Алгоритм роботи підпрограми перевірки отриманого протоколу

Алгоритм підпрограми виконує дуже важливу роль у визначенні протоколу який використовується клієнтом. А саме PAP або CHAP, в загальному ці протоколи мають схожий характер перевірки.

Розглянемо алгоритм роботи підпрограми, а саме усіх блоків:

1. **Перевірка протоколу.** На цьому етапі система визначає, який саме протокол буде використовуватись для автентифікації клієнта:

- Порівнюються параметри запиту з підтримуваними протоколами (наприклад, PAP, CHAP, MS-CHAP, EAP).

- Якщо протокол не підтримується, процес завершується з помилкою.

2. **Ініціалізація протоколу**. Після вибору протоколу виконується його ініціалізація:

- Генеруються необхідні параметри для роботи протоколу (наприклад, запит на пароль, генерація виклику тощо).

- Перевіряється відповідність протоколу специфікації для поточного запиту.

3. **CHAP.** На цьому етапі система визначає, чи використовується протокол CHAP:

- Якщо використовується CHAP, запускається процес хешування пароля.

- У разі невідповідності протоколу CHAP система переходить до іншого способу обробки (наприклад, PAP або MS-CHAP).

4. **Отримання пароля в базі даних.** Для автентифікації через CHAP необхідно отримати збережений пароль клієнта:

- Виконується запит до бази даних, де зберігаються облікові записи.

- Якщо пароль не знайдено, запит завершується відправкою сигналу reject.

5. **Створення хеша пароля.** Якщо пароль клієнта знайдено:

- Генерується хеш пароля на основі виклику (challenge), який було створено на етапі ініціалізації протоколу.

- Хешування забезпечує захист пароля під час передачі в мережі.

6. **Отримання пароля.** Після хешування виконується перевірка:

- Якщо згенерований хеш відповідає очікуваному, запит успішний і клієнт авторизується.

- У разі невідповідності — надсилається сигнал reject.

7. Пароль отримано. У разі отримання пароля клієнта (для інших протоколів, таких як PAP):

- Система перевіряє, чи пароль відповідає збереженому в базі.

- Якщо відповідь позитивна, продовжується створення особистого ключа клієнта.

8. **Створення особистого ключа клієнта.** Для додаткової безпеки створюється унікальний ключ:

- Ключ може ґрунтуватися на параметрах клієнта (наприклад, MAC-адресі, часу підключення).

- Цей ключ використовується для майбутньої ідентифікації сесії.

9. **Хешування ключа.** Створений ключ хешується для захисту:

- Використовуються сучасні алгоритми (наприклад, SHA-256) для забезпечення безпеки даних.

- Хешоване значення зберігається для подальшого порівняння.

10. **Збереження ключа.** Після хешування ключ зберігається у базі даних:

- Це необхідно для перевірки у подальших сесіях або при повторній автентифікації.

- Дані зберігаються у безпечному форматі.

11. **Відправка сигналу reject.** Якщо на будь-якому етапі відбувається помилка (наприклад, невідповідність пароля, відсутність клієнта в базі даних), система завершує процес і надсилає клієнту сигнал reject:

- Цей сигнал повідомляє клієнту про неможливість завершення автентифікації.

- Процес припиняється.

Ця схема ілюструє процес автентифікації клієнта з урахуванням використання різних протоколів, забезпечуючи як безпеку, так і надійність у системі.

4.2 Захист розробленого програмного забезпечення

Захист програмного забезпечення є критично важливим етапом розробки, особливо якщо йдеться про системи автентифікації, такі як freeradiusd3 для FreeBSD. Ці системи обробляють чутливі дані, як-от облікові записи користувачів і мережеві ключі, тому вразливості можуть призвести до серйозних наслідків, включаючи компрометацію даних або втрату контролю над інфраструктурою.

У цьому розділі розглянуто основні аспекти захисту freeradiusd3, включаючи безпечну конфігурацію, захист від зовнішніх атак і використання механізмів підвищення безпеки.

Основні підходи до захисту

Безпечне програмування:

- уникнення вразливостей, пов'язаних із переповненням буфера та некоректним керуванням пам’яттю;

- використання інструментів статичного аналізу коду для виявлення потенційних помилок.

Контроль доступу:

- обмеження прав доступу до файлів конфігурації та баз даних;

- використання моделей контролю доступу, таких як MAC (Mandatory Access Control), у FreeBSD.

Шифрування даних:

- захист конфіденційної інформації, як-от паролі, ключі та сертифікати, за допомогою сучасних алгоритмів шифрування;

- підтримка протоколів TLS для шифрування переданих даних.

Захист конфігурації FreeRADIUS:

Обмеження доступу до конфігураційних файлів:

- налаштування прав доступу таким чином, щоб тільки користувачі з адміністративними привілеями могли змінювати ці файли;

- використання механізмів, таких як chflags у FreeBSD, для встановлення атрибутів "тільки для читання".

Шифрування секретів:

- зберігання паролів і секретів у зашифрованому вигляді;

- впровадження механізмів захисту від витоку інформації через конфігурацію.

Верифікація сертифікатів:

- автоматизована перевірка SSL/TLS-сертифікатів, які використовуються для автентифікації серверів і клієнтів.

Захист від атак:

- запобігання атакам типу "зловмисне повторення" (Replay Attacks);

- використання одноразових токенів або часових обмежень для сесій.

Захист від DDoS-атак:

- впровадження обмеження швидкості запитів за допомогою брандмауера pf у FreeBSD;

- використання систем виявлення вторгнень (IDS), як-от Snort або Suricata.

Моніторинг аномальної активності:

- налаштування журналювання активності RADIUS-сервера для виявлення підозрілих дій;

- інтеграція з системами SIEM для централізованого аналізу безпеки.

Тестування безпеки:

- імітація великого навантаження на сервер для перевірки його стійкості до атак, пов’язаних із перевантаженням.

Penetration Testing:

- проведення тестів на проникнення для виявлення слабких місць у freeradiusd3;

- використання таких інструментів, як Metasploit або Nessus.

Періодичне оновлення:

- забезпечення регулярного оновлення freeradiusd3 для закриття відомих вразливостей.

Захист freeradiusd3 є багатогранним завданням, яке потребує уваги до кожного етапу розробки та впровадження. Поєднання безпечного програмування, використання інструментів FreeBSD та сучасних методів кіберзахисту забезпечить надійність і стійкість системи до атак.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Впровадження системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS є важливим етапом, що вимагає ретельного планування та виконання. Процес впровадження може бути розділений на кілька ключових етапів, кожен з яких включає в себе специфічні завдання та дії.

Перед початком безпосереднього впровадження системи необхідно провести попередню підготовку. Цей етап включає:

1. Аналіз вимог: Визначення потреб бізнесу та специфікацій, які повинні бути реалізовані в системі автентифікації. Це може включати типи автентифікації (PAP, CHAP, EAP тощо), кількість користувачів, які будуть автентифікуватися, а також особливі вимоги до безпеки.

2. Оцінка інфраструктури: Аналіз існуючої мережевої інфраструктури для виявлення можливих обмежень та потреб у оновленнях. Це може включати перевірку апаратного та програмного забезпечення, а також забезпечення необхідної пропускної здатності.

3. Вибір обладнання та програмного забезпечення: Вибір апаратного забезпечення (сервери, маршрутизатори, точки доступу) та програмного забезпечення, яке буде використовуватися для реалізації FreeRADIUS, включаючи вибір операційної системи та супутніх компонентів (бази даних, системи білінгу тощо).

Розробка та налаштування системи.

На цьому етапі відбувається безпосередня розробка та налаштування системи автентифікації:

1. Встановлення FreeRADIUS: Встановлення та налаштування FreeRADIUS на сервері. Це передбачає налаштування конфігураційних файлів для підтримки обраних методів автентифікації.

2. Налаштування бази даних: Інтеграція FreeRADIUS з обраною базою даних (MySQL, PostgreSQL тощо) для зберігання інформації про користувачів, їх права доступу та історію підключень.

3. Конфігурація політик доступу: Розробка та налаштування політик доступу, які визначають правила автентифікації та авторизації для різних груп користувачів.

4. Інтеграція з білінговою системою: Налаштування зв'язку між FreeRADIUS та білінговою системою для автоматизації процесу обліку та виставлення рахунків за використання ресурсів.

Тестування системи.

Після налаштування системи важливо провести комплексне тестування, щоб виявити та усунути можливі помилки:

1. Функціональне тестування: Перевірка всіх функцій системи, включаючи автентифікацію, авторизацію та облік. Тестування повинно включати різні сценарії, такі як успішна та неуспішна автентифікація, обробка помилок тощо.

2. Тестування продуктивності: Оцінка продуктивності системи під навантаженням, що дозволяє визначити максимальну кількість одночасних підключень та швидкість обробки запитів.

3. Тестування безпеки: Проведення тестів на вразливість, щоб виявити можливі загрози безпеці та забезпечити захист даних користувачів.

Навчання персоналу.

Важливим аспектом впровадження системи є підготовка користувачів та технічного персоналу:

1. Навчання адміністрації системи: Проведення тренінгів для адміністрації, які забезпечать їх знанням про налаштування, моніторинг та обслуговування FreeRADIUS.

2. Підготовка користувачів: Інформування кінцевих користувачів про нові процедури підключення до мережі, способи автентифікації та підтримка у випадку проблем.

Впровадження в експлуатацію.

Після завершення всіх попередніх етапів система готова до впровадження в промислову експлуатацію:

1. Перехід на продуктивний режим: Останнє тестування перед запуском в експлуатацію та переведення системи в режим реального часу.

2. Моніторинг системи: Після запуску важливо забезпечити моніторинг роботи системи, виявлення можливих проблем та швидке реагування на них.

3. Постійне вдосконалення: Збір зворотного зв'язку від користувачів і адміністрації для вдосконалення системи, впровадження нових функцій та оновлень.

Впровадження системи автентифікації користувачів на основі FreeRADIUS вимагає комплексного підходу та злагодженої роботи всіх етапів.

З правильним плануванням та реалізацією система може забезпечити високий рівень безпеки, зручність для користувачів та ефективність у роботі. Успішне впровадження також дозволяє організаціям ефективно управляти доступом до своїх ресурсів, забезпечуючи контроль і облік використання.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи автентифікації користувачів в інтернет-мережах на основі сервера FreeRADIUS.

Метою розробки є дослідження та програмна реалізація системи автентифікації користувачів з використанням FreeRADIUS для забезпечення безпечного доступу до інтернет-ресурсів.

Об'єктом дослідження є процеси автентифікації, авторизації та обліку (AAA) в сучасних мережах.

Предметом дослідження є сервер FreeRADIUS як інструмент реалізації систем централізованої автентифікації та управління доступом.

Методи дослідження включають:

- аналіз ефективності протоколів аутентифікації (EAP-TTLS, EAP-PEAP);

- дослідження методів інтеграції FreeRADIUS із зовнішніми базами даних (SQL та NoSQL);

- тестування продуктивності та навантаження на сервер автентифікації;

- дослідження методів забезпечення безпеки та захисту від атак.

Наукова новизна отриманих результатів. У процесі вирішення завдань, визначених цілями дослідження, отримано наступні результати:

- досліджено можливості використання FreeRADIUS як ефективного інструменту для реалізації централізованих систем автентифікації, зокрема для масштабованих інтернет-мереж;

- реалізовано систему автентифікації користувачів із використанням сучасних протоколів EAP-TTLS та EAP-PEAP для забезпечення захищеної передачі облікових даних;

- розроблено рівень взаємодії з базами даних на основі SQL (MySQL) та NoSQL (MongoDB) для динамічного управління обліковими записами та політиками доступу;

- здійснено інтеграцію з мережею через підтримку протоколу RADIUS для забезпечення гнучкої взаємодії з різними точками доступу та VPN-сервісами;

- проведено тестування продуктивності сервера FreeRADIUS у сценаріях високого навантаження, що дозволило визначити оптимальні параметри конфігурації для підвищення стійкості до перевантажень;

- розроблено й впроваджено додаткові механізми захисту системи, включаючи аудит подій та моніторинг підозрілих дій із використанням спеціалізованого програмного забезпечення.

Ці результати дозволяють покращити ефективність та безпеку систем автентифікації, що базуються на FreeRADIUS, та сприяють розвитку інструментів управління доступом у сучасних інтернет-мережах.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та програмної реалізації системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS можуть бути цікавими для різних груп осіб і організацій. Зведене по категоріях подане на рисунку 7.1.

1. ІТ-компанії та постачальники послуг	**Постачальники послуг інтернету (ISP) (**можуть впровадити систему автентифікації для управління доступом до своїх мереж);
	Хостинг-компанії (зацікавлені у надійних методах автентифікації для захисту даних користувачів).
2. Організації, що займаються кібербезпекою	**Консалтингові фірми з кібербезпеки** (можуть використовувати результати для розробки рекомендацій з безпеки для своїх клієнтів);
	Дослідники в області безпеки (зацікавлені у вивченні нових методів захисту даних і автентифікації).
3. Розробники програмного забезпечення	**Програмісти та системні адміністратори (**можуть використовувати отримані результати для створення або вдосконалення своїх власних систем автентифікації);
	Інженери з безпеки (зацікавлені в оптимізації існуючих систем та розробці нових рішень на базі FreeRADIUS).
4. Навчальні заклади(	**Університети та технічні коледжі (**можуть використовувати результати для навчальних програм з інформаційних технологій і безпеки);
	Дослідницькі лабораторії (зацікавлені в проведенні подальших досліджень на базі отриманих даних).
5. Користувачі та споживачі	**Кінцеві користувачі (**знайомі з системами автентифікації, можуть бути зацікавлені в розумінні, як їхні дані захищаються);
	Організації, що здійснюють обробку персональних даних (можуть використовувати результати для забезпечення відповідності вимогам з захисту даних).
6. Державні організації та регулятори	**Владні установи (**зацікавлені у впровадженні надійних систем автентифікації для захисту державних інформаційних систем);
	Регулятори в галузі захисту даних (можуть використовувати результати для розробки політик і рекомендацій).

Рисунок 7.1 – Цільова аудиторія

Отже, результати дослідження та програмної реалізації системи автентифікації на основі FreeRADIUS можуть бути корисними для широкого кола осіб і організацій, включаючи бізнес, наукові установи, навчальні заклади, державні організації та споживачів. Це може сприяти підвищенню безпеки та ефективності у сфері інформаційних технологій.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Оцінка привабливості програмної реалізації системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS може бути виконана шляхом застосування експертних оцінок.

Перед початком експертного оцінювання важливо визначити критерії, за якими буде оцінюватися привабливість системи. Для проекту критеріями можуть бути (рисунок 7.2).

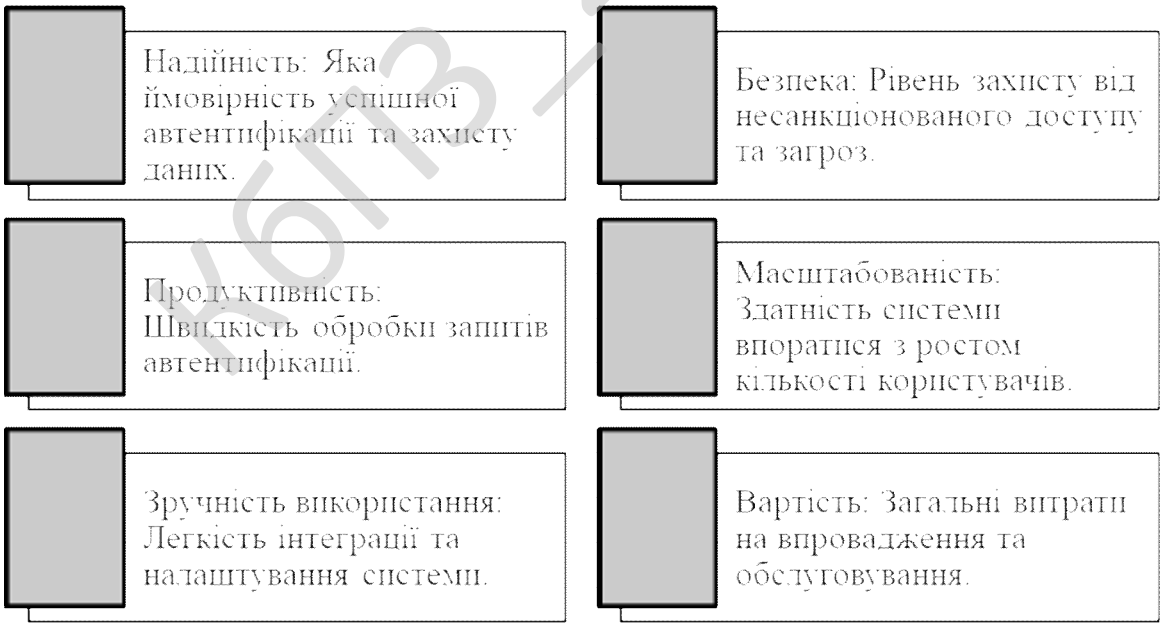

Рисунок 7.2 – Критерії експертної оцінки

Збираємо команду експертів у галузі інформаційних технологій, кібербезпеки та управління проектами. До експертів включаємо: спеціалістів з без-

пеки, системних адміністраторів, розробників програмного забезпечення, менеджерів проектів.

Використовуємо метод анкетування для збору оцінок від експертів. Кожен експерт оцінює зазначені критерії за шкалою, наприклад, від 1 до 5, де: 1 — дуже погано; 2 — погано; 3 — задовільно; 4 — добре; 5 — відмінно.

Після обробки отриманих даних, проводимо обрахунок середнього значення оцінок за кожним критерієм.

Таблиця 7.1 – Зведені результати експертних оцінок

Критерій	**Експерт 1**	**Експерт 2**	**Експерт 3**	**Середня оцінка**
Надійність	4	5	4	4.33
Безпека	5	4	5	4.67
Продуктивність	4	4	3	3.67
Масштабованість	4	5	4	4.33
Зручність	3	4	3	3.33

Далі визначаємо загальну привабливість системи, обчисливши зважене середнє оцінок за критеріями на основі їхньої важливості. Попередньо присвоюємо кожному критерію вагу (у відсотках): надійність: 25%; безпека: 25%; продуктивність: 15%; масштабованість: 15%; зручність: 10%; вартість: 10%.

Загальна оцінка =(0.25*4.33) + (0.25*4.67) + (0.15*3.67) + (0.15*4.33) + (0.10*3.33)+(0.10*3.67)

Загальна оцінка в даному випадку дорівнює 4.15. Це свідчить про високу привабливість проєкту, що може бути позитивним фактором для прийняття рішення щодо його реалізації.

Цей підхід до оцінки привабливості системи автентифікації на основі FreeRADIUS за допомогою експертних оцінок дозволяє отримати об'єктивні

дані, які можуть допомогти в прийнятті рішень щодо впровадження та розвитку системи.

7.3 Вибір методу оцінки вартості ПЗ

Для програмної реалізації системи автентифікації на основі FreeRADIUS оптимальним підходом може бути поєднання методів. Можна почати з методу витрат для детальної оцінки витрат, а також використати метод аналогів для перевірки відповідності на ринку. Додатково, застосування функціональних точок може допомогти в отриманні об'єктивних оцінок на основі вимог до системи.

Варто пам’ятати, що кожен метод має свої переваги. Метод аналогів (порівняння з ринковими цінами) передбачає дослідження цін на подібні рішення на ринку. Порівнюючи з конкурентами або аналогічними системами, можна оцінити вартість розробки та впровадження вашого рішення. Ключовою перевагою є швидкість і простота в оцінці, можливість отримати актуальну інформацію про ринок.

Метод витрат – це оцінка вартості на основі витрат, пов'язаних із розробкою, впровадженням та підтримкою системи. Включає витрати на персонал (зарплата розробників, тестувальників тощо), обладнання, ліцензії та інші витрати. Дозволяє детально прорахувати всі витрати, що забезпечує точність.

Також, важливо пам’ятати, що залучення експертів може допомогти в отриманні більш точної оцінки вартості, особливо в умовах невизначеності.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Таблиця 7.2 – Основні показники впровадження проєкту

1. **Зниження витрат на безпеку** **Зменшення витрат на обслуговування**: Використання FreeRADIUS дозволяє знизити витрати на ліцензування програмного забезпечення Витрати на ліцензії альтернативних рішень: 10,000 USD на рік. Витрати на FreeRADIUS: 0 USD. **Економія**: 10,000 USD на рік.
2. **Підвищення продуктивності** **Швидкість обробки запитів**: Завдяки оптимізації, система на основі FreeRADIUS може обробляти запити автентифікації швидше Скорочення часу обробки запиту з 5 секунд до 2 секунд. При 10,000 запитах на день, економія часу: 30,000 секунд (8.33 години) на день. **Економія на витратах на підтримку**: 1,000 USD на місяць (за рахунок зменшення навантаження на технічну підтримку).
3. **Збільшення кількості користувачів** **Задоволеність клієнтів**: Впровадження швидшої та надійної системи автентифікації може підвищити рівень задоволеності клієнтів. Приріст нових користувачів: 20% (додатково 2,000 користувачів на рік). Середній дохід від одного користувача: 100 USD на рік. **Додатковий дохід**: 200,000 USD на рік.
4. **Покращення безпеки** **Зменшення витрат на реагування на інциденти**: Підвищення рівня безпеки призводить до зменшення інцидентів, пов'язаних із несанкціонованим доступом. Зменшення інцидентів: з 10 до 2 на рік. Витрати на реагування на інциденти (втрати): 5,000 USD на інцидент. **Економія**: 40,000 USD на рік.
5. **Оцінка загальної економічної ефективності** Обчислимо загальну економічну ефективність, підсумувавши всі вигоди:

Показник	**Сума (USD)**
Економія на ліцензіях	10,000
Економія на витратах на підтримку	1,000 * 12 = 12,000
Додатковий дохід від нових користувачів	200,000
Економія на реагуванні на інциденти	40,000
Загальна економічна вигода	**262,000**

Впровадження системи автентифікації користувачів на основі FreeRADIUS демонструє значну економічну ефективність через зниження витрат, підвищення продуктивності, збільшення доходів від нових користувачів та зменшення витрат на безпеку. Таким чином, загальна економічна вигода складає 262,000 USD на рік, що свідчить про доцільність впровадження даної системи.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Алгоритм просування проєкту програмної реалізації системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS зведено до рисунку 7.3.

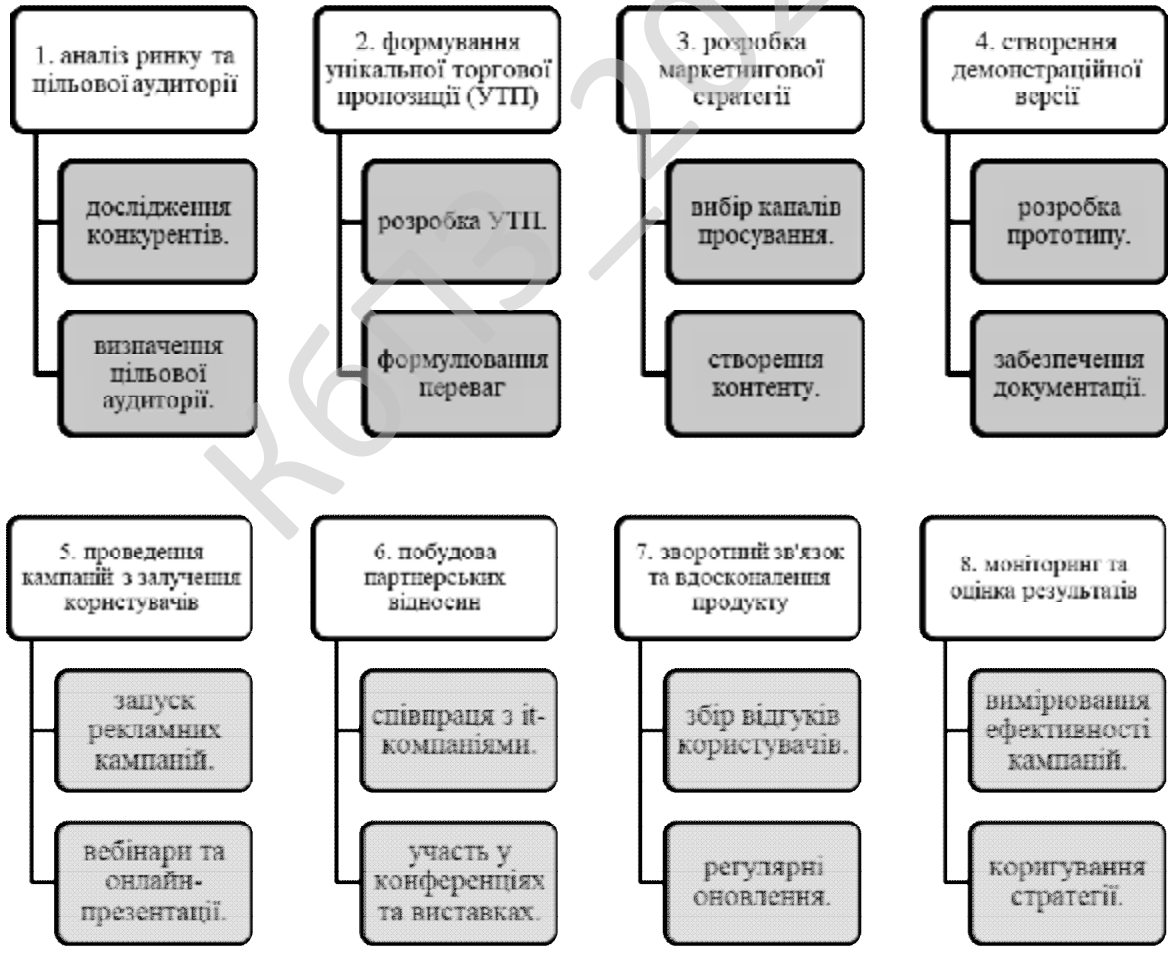

Рисунок 7.3 – Алгоритм просування проєкту

Цей алгоритм допоможе ефективно просувати проєкт програмної реалізації системи автентифікації користувачів на основі FreeRADIUS, залучаючи нових користувачів та створюючи позитивний імідж продукту на ринку.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації каналів збуту та шляхів реалізації проєкту програмної реалізації системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS варто розглянути пропозиції, наведені на рисунку 7.4.

Оптимізація каналів збуту та шляхів реалізації проєкту програмної реалізації системи автентифікації користувачів на основі FreeRADIUS вимагатиме комплексного підходу, що включає використання онлайн-платформ, партнерських відносин, активне просування через соціальні мережі та участь у професійних заходах. Це допоможе залучити нових клієнтів та забезпечити успіх продукту на ринку.

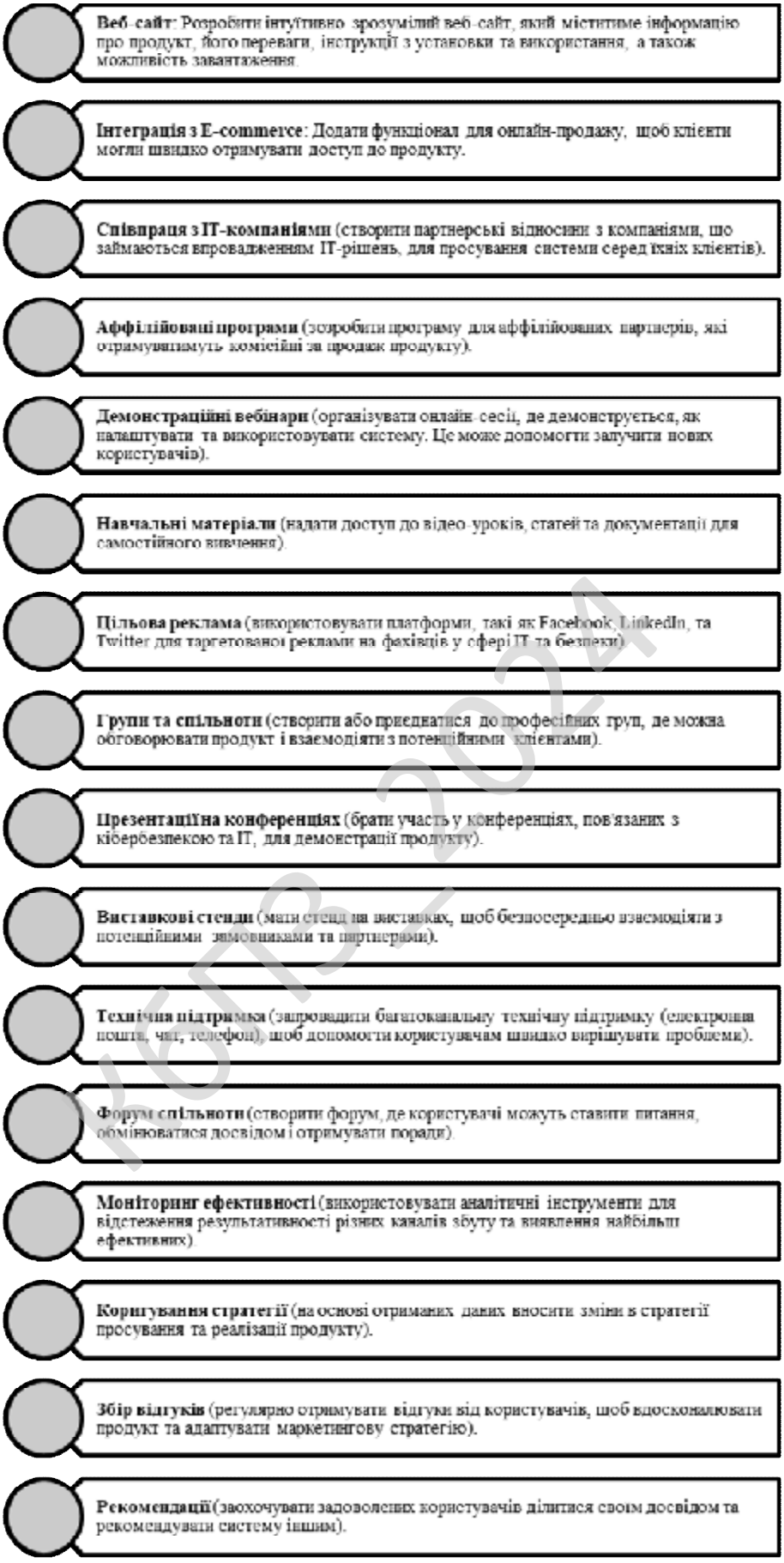

Рисунок 7.4 – Ключові фактори успіху проєкту

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключові фактори успіху проєкту програмної реалізації системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS можуть фактори з рисунку 7.5.

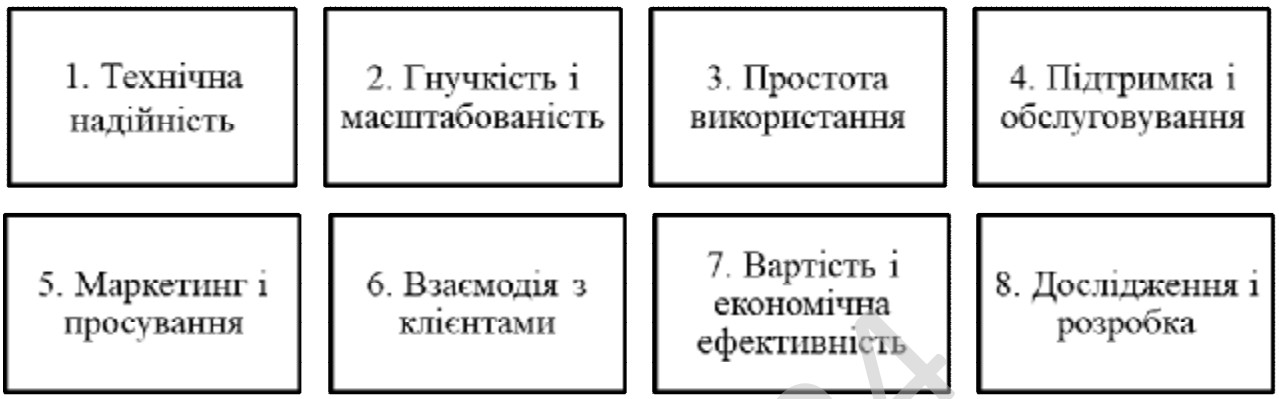

Рисунок 7.5 – Ключові фактори успіху проєкту

Успіх проєкту реалізації системи автентифікації користувачів на основі FreeRADIUS залежить від багатьох факторів, які включають технічні аспекти, взаємодію з клієнтами, ефективний маркетинг та постійне вдосконалення продукту. Зосереджуючи увагу на цих факторах, можна значно підвищити шанси на успіх проєкту.

8. ЗАХОДИ ЩОДО ОХОРОНИ ПРАЦІ І ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здо-

ров'я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

- параметри повітряного середовища в приміщенні;
- вентиляція приміщення;
- освітлення приміщення;
- параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машин (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

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

- ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.
- ризик виникнення пожежі;
- негативний вплив на органи зору людини;
- ризики ураження електричним струмом;
- недостатня, або надмірна освітленість робочого місця;
- електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

- несприятливі мікрокліматичні умови;

- нервово-емоційна напруженість праці;

- інтелектуальні навантаження;

- монотонність праці;

- невідповідність ергономічних показників робочого місця діючим вимогам;

- шум;

- статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 - Розміри приміщення

Найменування	Значення, м
Ширина	3
Довжина	4,6
Висота	3

Таблиця 8.2 - Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,9
Об'єм, V	$м^3$	не менше 20.0	20,7

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працюють двоє людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [5], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація.

Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості

руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 - Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Вологість, %	Швидкість повітря, м/с	Температура,°С	Вологість %	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	40-55	0,12
Тепла	23-25	50-70	0,1	24-25	50-65	0,9

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер Xerox WorkCentre 3025BI (3025VBI), електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми україни "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В).

Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню.

Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1],

Крім того все поле зору повинне бути освітлено достатньо рівномірно - ця основна гігієнічна вимога.

Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

- мікроклімат відповідає нормативному значенню;

- акустичні умови роботи не перевищують нормативних значень;

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

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

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

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

8.5 Розрахункова частина

Проводемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 3 м, довжина – 4,6 м, висота – 3 м.

У зазначеному приміщенні працює 1 особа.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де: F - світловий потік, що розраховується, Лм;

E - нормована мінімальна освітленість, Лк; E = 300 Лк;

S - площа освітлюваного приміщення (у нашому випадку S=3 х 4,6 = 13,8 м^2);

K - коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5);

Z - відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1);

n - коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де: S - площа приміщення, S = 13,8 м^2;

h - розрахункова висота підвісу, h = 3 м (співпадає з висотою стелі, т.я. лампи освітлення закріплюються на стелі);

A - ширина приміщення, A = 3 м;

B - довжина приміщення, B = 4,6 м.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$i=0,43.$$

Знаючи індекс приміщення, за знаходимо $n = 0,23$ (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом ламп). Підставимо всі значення у формулу, визначемо світловий потік: *F=29700 Лм.*

Для розрахунку дудемо використовувати світлодіодні панелі *LED панель 42Вт 6000К SUNLED 000000127*, світловий потік яких $F_л = 3990$ *Лм.*

Число ламп визначається по формулі:

$$N=F/F_л$$

де: F - світловий потік,

$F_л$ - світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$N= 29700/ 3990=7,4 \text{ шт.}$$

Приймаємо необхідну кількість світлодіодних світильників 8 шт.

Висновки до розділу

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

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

У результаті виконання даного дипломного магістерського проекту була розроблена та впроваджена система автентифікації користувачів в інтернет-мережах на основі технології FreeRADIUS. Основною метою проекту стало забезпечення надійної, безпечної та ефективної автентифікації для користувачів, що підключаються до мережі.

У процесі дослідження були розглянуті ключові аспекти функціонування системи, включаючи етапи автентифікації, авторизації та обліку, що є основними компонентами AAA (Authentication, Authorization, Accounting). Особлива увага була приділена методам автентифікації, які можуть бути реалізовані в системі, зокрема PAP, CHAP та EAP, що забезпечує гнучкість у налаштуванні безпеки доступу.

На етапі розробки та впровадження системи були виконані комплексні тестування для перевірки її працездатності та надійності під навантаженням, що дозволило виявити та усунути можливі недоліки. Також була розроблена інтеграція з білінговою системою, що дозволяє автоматизувати процес обліку використання ресурсів та спрощує управління фінансовими операціями.

Результати впровадження системи підтвердили її ефективність у забезпеченні безпеки та зручності для користувачів. Нова система дозволяє швидко та безпечно підключати клієнтів до мережі, а також забезпечує можливість моніторингу та управління доступом у реальному часі.

Отже, завершення даного проекту стало важливим кроком у напрямку підвищення безпеки інформаційних систем та забезпечення надійності їх роботи. Подальші етапи можуть включати розширення функціоналу системи, вдосконалення механізмів захисту та адаптацію до нових вимог бізнесу та технологій.

Цей проект демонструє можливості FreeRADIUS як потужного інструменту для реалізації систем автентифікації в умовах сучасних інформаційних технологій, що відкриває нові горизонти для їх застосування в різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Andrew Findlay. FreeRADIUS Beginner's Guide. - 2012. -138c.

2. Офіційна документація FreeRadius - [Електронний ресурс] - Режим доступу: https://www.freeradius.org/

3. Офіційна документація Perl - [Електронний ресурс] - Режим доступу: https://www.freeradius.org/

4. Офіційна документація Python - [Електронний ресурс] - Режим доступу: https://www.freeradius.org/

5. Larry Wall., Tom Christiansen, Jon Orwant Programming Perl. -2012. - 1312c

6. Eric Matthes. Python Crash Course. - 2015. -544 c.

7. Наукова стаття RADIUS: Remote Authentication Dial In User Service [Електронний ресурс] - Режим доступу: https://ieeexplore.ieee.org/Xplore/home.jsp

8. Наукова стаття Understanding 802.1X: The Network Access Control [Електронний ресурс] - Режим доступу: https://www.techtarget.com/query?802.1X

9. Наукова стаття A Survey of PPPoE (Point-to-Point Protocol over Ethernet) - [Електронний ресурс] - Режим доступу: https://www.researchgate.net/

10. Наукова стаття IP over Ethernet (IPoE): A Comprehensive Overview - [Електронний ресурс] - Режим доступу: https://www.cisco.com/c/en/ios-ip-over-ethernet-xe-16-book.pdf

11. Korry Douglas. PostgreSQL: A Comprehensive Guide to Building, Programming, and Administering PostgreSQL Databases. – 2019. – 600 c.

12. Christophe Pettus, Simon Riggs. PostgreSQL 11 Administration Cookbook. – 2019. – 500 c.

13. Greg Smith. PostgreSQL 9.0 High Performance. – 2010. – 450 c.

14. Tom Lane, Bruce Momjian. PostgreSQL: Introduction and Concepts. – 2001. – 456 c.

15. Офіційна документація PostgreSQL – [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/

16. Larry Wall, Randal L. Schwartz, Tom Christiansen. Programming Perl. – 2012. – 1136 с.

17. Simon Cozens. Advanced Perl Programming. – 2005. – 304 с.

18. Damian Conway. Perl Best Practices. – 2005. – 542 с.

19. Brian J. Trammell. FreeRADIUS Beginner's Guide. – 2011. – 360 с.

20. Офіційна документація Juniper Networks – [Електронний ресурс] – Режим доступу: https://www.juniper.net/documentation/

21. Douglas Richard Hanks Jr. Juniper MX Series. – 2016. – 700 с.

22. Harry Reynolds. JUNOS Enterprise Routing. – 2007. – 600 с.

23. Peter Southwick. Juniper SRX Series. – 2013. – 800 с.

24. Nikhil Swaminathan. Authentication and Authorization for the Internet of Things – 2017. – 200 с.

25. Scientific Article: RADIUS Authentication for Internet Access – [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/

26. Scientific Article: Authorization and Access Control in IoT Networks – [Електронний ресурс] – Режим доступу: https://www.researchgate.net/

27. Scientific Article: RADIUS Protocol and Network Security – [Електронний ресурс] – Режим доступу: https://www.techtarget.com/

28. Державні будівельні норми України: ДБН В.2.5-28:2018. - Режим доступу до ресурсу: https://goo.su/9AkQ

29. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

30. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

31. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

32. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

33. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. - Кіровоград: Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

34. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення 19.09.22).

35. Сакулін В.П., Шептовицький В.М. Безпека праці під час монтажу та експлуатації електроустановок / В.П. Сакулін, В.М. Шептовицький. – Л. : "Колос", 1973. – 238 с.

36. Центр післядипломної освіти та підвищення кваліфікації. - Режим доступу до ресурсу: https://cpo.stu.cn.ua

37. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. - 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

38. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. - Кропивницький: ЦНТУ, 2022. — 19 с. [Електронний ресурс]. – Режим доступу : http://dspace.kntu.kr.ua/jspui/handle/123456789/12240

39. WebSocket API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

40. Android USB Host and Accessory [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/connectivity/usb

41. Bluetooth overview for Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/connectivity/bluetooth

42. Android Canvas and Drawing [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/graphics/2d-graphics

43. Windows Input Injection API [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/api/_input_injector/

44. Android Network Security Configuration [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/training/articles/security-config

45. Android Developer Documentation: Input Events Overview [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/ui/ui-events

46. Windows Developer Documentation: Windows Ink [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/apps/design/input/pen-and-stylus-interactions

47. WebSocket Protocol [Електронний ресурс] – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc6455

48. Android Developer Documentation: Wi-Fi P2P [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topics/connectivity/wifip2p

49. Android Developer Documentation: USB host and accessory [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/connectivity/usb

50. Windows Developer Documentation: USB [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 07.08.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) 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;

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

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS;

– цілісність даних у процесі роботи, обробки 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.

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 архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

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.

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

Perl.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати маркетингове та економычне обґрунтування ІТ-проєкту з урахуванням цін на 3 вересня 2024 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці на робочому місці ІТ-фахівця.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 62 аркуші.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 02.12.2024 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист __.__.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник бакалаврської дипломної роботи

_____ Якименко Н. М.

Дослідження та програмна реалізація системи автентифікації користувачів в інтернет-мережах на основі FreeRADIUS

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 23

Літера: РП

Кропивницький – 2024 року

Створення загальних методів авторизації та мехінизму автентифікації

```
server {

    authorize {
        preprocess
        files
        expiration
        logintime
        perl
    if (&request:Certain-Method == 'PPPoE') {
        pap
      chap
      mschap
    } else {
        update control {
            Auth-Type := Perl
        }
      }
    }

    authenticate {
        Auth-Type PAP {
            pap
        }
        Auth-Type CHAP {
            chap
        }
        Auth-Type MS-CHAP {
            mschap
        }
        Auth-Type Perl {
            perl
        }
    }

    preacct {
        preprocess
        acct_unique
        files
    }

    accounting {
        perl
    }

    session {
    }

    post-auth {
        files
        perl
        Post-Auth-Type REJECT {
            attr_filter.access_reject
        }
    }

    pre-proxy {
    }

    post-proxy {
    }
}
```

Релізація протоколу відключення активного клієнта

```
use Redis;
use JSON;
use Net::Radius::Client;
use settings;
use Data::Dumper;
use Sys::Syslog qw(:standard :macros);

sub init_log {
    my $logdir = $settings::LOGDIR;
    my $log_destination = $settings::LOG_DESTINATION;
    if ($log_destination eq 'syslog') {
        openlog('radiusdestroy', 'ndelay,pid', LOG_DAEMON);
    }
}

sub LOG {
    my $message = shift @_;
    syslog(LOG_INFO, $message);
}

init_log();

sub nakUser {
    my ($UserName, $NasIp, $Secret, $AcctSessionId) = @_;
    Net::Radius::Client::load("/usr/local/etc/raddb/dict/dictionary");
    my $servers = {
        $NasIp => {
            3799 => {
                'secret' => $Secret,
                'timeout' => 2,
                'retries' => 1
            }
        }
    };

    my $req = {
        0 => {
            'User-Name' => [$UserName],
            'NAS-IP-Address' => [$NasIp],
            'Acct-Session-Id' => [$AcctSessionId]
        }
    };
    my ($code, $rsp) = Net::Radius::Client::query($servers, "Disconnect-
Request", $req);
    sleep 2;
    LOG("(_nakUser)[$UserName] shutdown session returned $code.\n");
}

my $redis = Redis->new(
    server => "$settings::REDIS_HOST:$settings::REDIS_PORT",
    password => $settings::REDIS_PASSWORD,
);

$redis->select($settings::REDIS_DB_DESTROY);

LOG("Connected to Redis on host $settings::REDIS_HOST, database
$settings::REDIS_DB_DESTROY.\n");

LOG("Listening to list '$settings::REDIS_KEY_DESTROY'...\n");

while (1) {
    # Blocking pop from the list
    my ($list, $key) = $redis->blpop($settings::REDIS_KEY_DESTROY, 0);
    if (defined $key) {
        LOG("Received key from '$list': $key\n");
```

```
        # Connect to the second database
        my $redis2 = Redis->new(
            server => "$settings::REDIS_HOST:$settings::REDIS_PORT",
            password => $settings::REDIS_PASSWORD,
        );

        $redis2->select($settings::REDIS_DB_SESSIONS);

        my $session_key = "session:$key";
        my $value = $redis2->get($session_key);
        if (defined $value) {
            # JSON parsing
            my $data = decode_json($value);

            # Extract necessary parameters
            my $UserName = $data->{UserName};
            my $RadiusIp = $settings::RADIUS_IP;
            my $NasIp = $data->{NASIPAddress};
            my $Secret = $settings::SECRET;

            my ($AcctSessionId) = $data->{AcctSessionId} =~ /:(\d+)$/;

            if (defined $AcctSessionId) {
                nakUser($UserName, $NasIp, $Secret, $AcctSessionId);
            } else {
                LOG("Failed to extract Acct-Session-Id from value: '$value'\n");
            }
        } else {
            LOG("Key '$session_key' does not exist in the database.\n");
        }
    }
}
```

Створення та видача ip адреси користувачу

```
use utf8;

package IPPOOL;

use HTTP::Tiny;
use Time::Piece;
use JSON::PP qw(encode_json decode_json);
use Scalar::Util qw/reftype/;
use Data::Dumper;
use Socket qw(inet_aton);
use strict;
use warnings;

sub inactive {
    my $IP = unpack('N', inet_aton(shift @_)) & unpack('N',
inet_aton('255.255.192.0'));')))
    foreach my $NET (unpack('N', inet_aton('255.255.192.0')), unpack('N',
inet_aton('255.255.192.0')), unpack('N', inet_aton('255.255.192.0')),
unpack('N', inet_aton('255.255.192.0')), unpack('N', inet_aton('255.255.192.0{
        if ($IP == $NET) { return 1; }
    }
    return 0;
}

sub private {
    my $IP = unpack('N', inet_aton(shift @_)) & unpack('N',
inet_aton('255.255.192.0'));
    foreach my $NET (unpack('N', inet_aton('100.84.0.0')), unpack('N',
inet_aton('100.80.0.0')), unpack('N', inet_aton('100.64.128.0')), unpack('N',
inet_aton('100.69.0.0')), unpack('N', inet_aton('100.70.0.0'))) {
        if ($IP == $NET) { return 1; }
    }
    return 0;
}

sub static {
    my $IP = unpack('N', inet_aton(shift @_)) & unpack('N',
inet_aton('255.255.254.0'));
    my $NET = unpack('N', inet_aton('255.255.192.0'));
    if ($IP == $NET) { return 1; }
    return 0;
}

sub allocate {
    my $NAS = shift @_;
    my $POOLS = shift @_;
    my $USER = shift @_;

    my $sqlLock = $main::DBa->prepare("SELECT GET_LOCK(?, 1) AS Result") or
main::LOG("(Authorize $NAS)[$USER] SQL error $DBI::errstr");
    my $sqlPool = $main::DBa->prepare("SELECT Address FROM AddressPool WHERE
Pool=? AND NAS=? AND Prefered='HOME' AND (Expiration < NOW() OR Tenant=?) ORDER
BY Tenant<>?, Expiration LIMIT 1") or main::LOG("(Authorize $NAS)[$USER] SQL
error $DBI::errstr");
    my $sqlUnlock = $main::DBa->prepare("SELECT RELEASE_LOCK(?)") or
main::LOG("(Authorize $NAS)[$USER] SQL error $DBI::errstr");

    my $sqlAddress;
    foreach my $POOL (@{$POOLS}) {
        $sqlLock->execute($POOL."-".$NAS);
        my $Lock = $sqlLock->fetchrow_hashref();
        $sqlLock->finish();
        if ($Lock->{'Result'} == 0) {
            main::LOG("(Authorize $NAS)[$USER] Unable to lock IP pool $POOL,
switching");
```

```
            next;
        }
        $sqlPool->execute($POOL, $NAS, $USER, $USER);
        if ($sqlPool->rows != 0) {
            $sqlAddress = $sqlPool->fetchrow_hashref();
            $main::DBa->do("UPDATE AddressPool SET Tenant='$USER',
Expiration=NOW() + INTERVAL 30 MINUTE WHERE Address='$sqlAddress->{'Address'}'")
or main::LOG("(Authorize $NAS)[$USER] SQL error $DBI::errstr");
        }
        $sqlPool->finish();
        $sqlUnlock->execute($POOL."-".$NAS);
        $sqlUnlock->finish();
        if (defined($sqlAddress)) {
            return $POOL, $sqlAddress->{'Address'};
        }
        main::LOG("(Authorize $NAS)[$USER] Pool $POOL empty");
    }
    return undef, undef;
}

sub extend {
    my $NAS = shift @_;
    my $USER = shift @_;
    my $ADDRESS = shift @_;
    my $INTERVAL = shift @_;
    if (!defined($INTERVAL)) { $INTERVAL = '30 MINUTE'; }
    my $ROWS = $main::DBa->do("UPDATE AddressPool SET Expiration=NOW() +
INTERVAL $INTERVAL WHERE Address='$ADDRESS' AND Tenant='$USER'") or
main::LOG("(Accounting $NAS)[$USER] SQL error $DBI::errstr");
    if ($ROWS != 1) {
        return 0;
    }
    return 1;
}

sub extend_new {
    my $NAS = shift @_;
    my $ADDRESS = shift @_;
    my $INTERVAL = shift @_;
    if (!defined($INTERVAL)) { $INTERVAL = '30 MINUTE'; }
    my $ROWS = $main::DBa->do("UPDATE AddressPool SET Expiration=NOW() +
INTERVAL $INTERVAL WHERE Address='$ADDRESS'") or main::LOG("(Accounting $NAS)
SQL error $DBI::errstr");
    if ($ROWS != 1) {
        return 0;
    }
    return 1;
}

return 1;
```

Обробка запиту на авторизацію

```
sub readctl {
    my $conn = shift @_;
    my $fn = shift @_;
    my $req = shift @_;
    my $strings = shift @_;
    my $pats = shift @_;

    if(!open(IN, $fn) && $req) {
        &expire($conn, "Cannot read $fn");
    }

    my $ln;
    while($ln = <IN> || shift @_) {
        # Parse the line and make sure the setting is legal.
        chomp $ln;
        $ln or next;
        $ln =~ /^([a-zA-Z]+)\s*(\=|\+\=)\s*(\S.*)?$/ or
            &expire($conn, "$fn:$.: Cannot parse");
        my ($name, $sym, $data) = ($1, $2, $3);
        exists $strings->{$name} || exists $pats->{$name} or
            &expire($conn, "$fn:$.: Unknown control variable $name.");
        if($sym eq '+=' && !exists $pats->{$name}) {
            &expire($conn, "$fn:$.: Value $name cannot be extended.");
        }

        # Update the hash.
        if(exists $pats->{$name}) {
            # Pattern.
            if($sym eq '=') { $pats->{$name} = ''; }
            while($data) {
                $pats->{$name} .= '|' if($pats->{$name});
                my $p;
                if($data =~ s|^/(.*?[^\\])/\s*||) {
                    # Just add a pattern to the list.
                    $p = $1;
                } elsif($data =~ s|^"(.*?[^\\])"\s*||) {
                    # A string specified.  Make a pattern for exact match.
                    $p = $1;
                    $p =~ s/([^a-zA-Z0-9])/\\$1/g;
                    $p = "^($p)\$";
                } else {
                    &expire($conn, "$fn:$.: Bad match item spec.");
                }

                # Check for syntax error in the pattern, then add it
                # to the whole match pattern.
                defined eval("'x' =~ /$p/") or
                    &expire($conn, "$fn:$.: Bad pattern $p");
                $pats->{$name} .= "($p)";
            }
        } else {
            # Just plain value.
            $strings->{$name} = $data;
        }

    }

    close(IN);
}

# Options.
my $norun = 0;
my $rcfile = ".webserver";
my $csync = undef;
GetOptions("n|norun" => \$norun, "r|rmtdir|server=s" => \$rcfile,
           "s|sync:s" => \$csync);
```

```
$rcfile =~ /^\./ or $rcfile = ".$rcfile";

# Read the top-level command file.
my %strings = ( sync => 0, host => 0, acct => 0, pwd => 0, cd => '');
my %pats = ( send => "(\\.html\$)", preserve => "", descend => "(.)",
             presdir => "", dontsend => "", dontdescend => "" );
readctl(0, $rcfile, 1, \%strings, \%pats, @ARGV);
if(defined $csync) {
    # From command line.
    if($csync eq '') { $csync = 1; }
    $strings{"sync"} = $csync;
}
if($strings{"sync"} == 1) { $strings{"sync"} == 'syncfile'; }

#foreach my $n(keys %strings) { print "strings{$n} = $strings{$n}\n"; }
#print "\n";
#foreach my $n(keys %pats) { print "pats{$n} = $pats{$n}\n"; }
#print "\n";

# Time offset.
my $syncoff = 0;

# Close and maybe exit.
my $was_err = 0;
sub expire {
    my $conn = shift @_;
    my $msg = shift @_;
    my ($fatal) = (@_, 1);

    my $err = '';
    if($conn) {
        $err = $conn->message();
    }
    chomp $err;
    $err =~ s/\.\s*$//;
    $msg =~ s/\!\!/$err/;
    print "$msg.\n";
    if($fatal) {
        $conn->quit() if $conn;
        exit(2);
    } else {
        $was_err = 1;
    }
}

# Read the curr directory on the server, and return a pair of hash references.
# The first is to a hash giving the name and mod date of each ordinary file,
# and the second gives the directories, with value 1.
sub rmt_contents {
    my $conn = shift @_;
    my %files = ();
    my %dirs = ();
    my $nttime = '\d\d\-\d\d\-\d\d\s+\d\d\:\d\d[AP]M';

    my @names = $conn->dir();
    int($conn->code() / 100) == 2 or
        expire($conn, "Remote list failed: !!");
    foreach my $l(@names) {
        chomp $l;
        my ($name, $isdir);
        if($l =~ /^[\-d]([\-r][\-w][\-xs]){2}[\-r][\-w][\-xt]\s/) {
            # Smells like Unix (inoring specials, etc.)
            my ($mode, $links, $uid, $gid, $size, $d1, $d2, $d3, $n) =
                split(/\s+/, $l, 9);
            $name = $n;
            $isdir = ($l =~ /^d/);
        } elsif($l =~ /^$nttime\s+(\<(DIR)\>\s+|\d+\s)(.*)$/) {
            # Smells NT
            $name = $3;
```

```
            $isdir = ($2 eq 'DIR');
        } else {
            # Can't figure it out, or its the wrong kind of thing.
            next;
        }
        if($isdir) {
            next if $name eq '.' || $name eq '..';
            $dirs{"$name"} = 1;
        } else {
            my $time = $conn->mdtm($name);
            $files{"$name"} = $time + $syncoff;
        }
    }
    return (\%files, \%dirs);
}

# Read the curr local directory, and return a pair of references.
# The first is to a hash giving the name and mod dat of each ordinary file,
# and the second is to an array of directory names in the current directory.
# The function ignores any file starting with ., and obeys the send,
# dontsend, descend and dontdescend patterns.  That means that files which
# don't match send, or do match dontsend, are ignored.  Likewise directories
# not matchind descend or matching dontdescend are not reported.
sub loc_contents {
    my $conn = shift @_; # Only for aborts.
    my $strref = shift @_;
    my $pref = shift @_;

    my %files = ();
    my %dirs = ();

    opendir(CD, '.') or expire($conn, "Directory open failed: $!");
    while(my $name = readdir(CD))
    {
        next if($name =~ /^\./);
        if(-f $name) {
            if(!$pref->{"send"} || $name !~ /$pref->{"send"}/) { next; }
            if($pref->{"dontsend"} && $name =~ /$pref->{"dontsend"}/) {
                next;
            }
            my ($dev,$ino,$mode,$nlink,$uid,$gid,$rdev,$size,$atime,$mtime)
                = stat($name);
            $files{"$name"} = $mtime;
        } elsif(-d $name) {
            if(!$pref->{"descend"} || $name !~ /$pref->{"descend"}/) {
                next;
            }
            if($pref->{"dontdescend"} &&
               $name =~ /$pref->{"dontdescend"}/) {
                next;
            }
            $dirs{"$name"} = 1;
        }
    }
    closedir(CD);
    return (\%files, \%dirs);
}

# Delete all contents of the current remote directory, with the given name.
sub delrmt {
    my $conn = shift @_;
    my $name = shift @_;

    # Get the contents.
    my ($files, $dirs) = rmt_contents($conn);

    # Get rid of the files.
    foreach my $fi(keys %$files) {
        print "Deleting $name$fi.\n";
```

```
        if(!$norun) {
            $conn->delete($fi) or
                expire($conn, "Delete $name$fi failed: !!", 0);
        }
    }

    # Get rid of the directories.
    foreach my $d(keys %$dirs) {
        $conn->cwd($d) or expire($conn, "cd $name$d failed: !!");
        delrmt($conn, "$name$d/");
        $conn->cdup() or expire($conn, "cd up failed: !!");
        print "Removing directory $name$d.\n";
        if(!$norun) {
            $conn->rmdir($d) or expire($conn, "rmdir $name$d failed: !!", 0);
        }
    }
}

# Synchronize.
sub sync {
    my $conn = shift @_;
    my $name = shift @_;
    my $fn = shift @_;
    my $strings = shift @_;
    my $pats = shift @_;

    # Get the contents.
    my ($rfiles, $rdirs) = rmt_contents($conn);
    my ($lfiles, $ldirs) = loc_contents($conn,$strings,$pats);

    #print "FRED $name:\n"; prfl($lfiles);
    #print "\n";
    #prfl($ldirs);
    #print "\n";

    # Go through the local files and put what needs putting.
    foreach my $lf(keys %$lfiles) {
        # See if it must be sent.
        if(!exists $rfiles->{$lf} || $rfiles->{$lf} < $lfiles->{$lf}) {
            # Old or missing...
            print "Sending $name$lf\n";
            if(!$norun) {
                if(!-r $lf) {
                    expire($conn, "File $lf is not readable", 0);
                } else {
                    $conn->put($lf) or
                        expire($conn, "Put $name$lf failed: !!", 0);
                }
            }
        }
        delete $rfiles->{$lf};
    }

    # If there are any remote files left, delete them.
    foreach my $rf(keys %$rfiles) {
        next if ($pats->{"preserve"} && ($rf =~ /$pats->{"preserve"}/));
        print "Deleting $name$rf.\n";
        if(!$norun) {
            $conn->delete($rf) or
                expire($conn, "Delete $name$rf failed: !!", 0);
        }
    }

    # Recur on the local directory names.
    foreach my $ld(keys %$ldirs) {
        # Create if needed.
        if(!$rdirs->{$ld}) {
            print "Creating $name$ld\n";
            if($norun) {
```

```
                print "Sending $name$ld subtree.\n";
                next;
            } else {
                if(!$conn->mkdir($ld)) {
                    expire($conn, "Create $name$ld failed: !!", 0);
                    next;
                }
            }
        }

        # Perform the recursion.
        $conn->cwd($ld) or expire($conn, "cd $name$ld failed: !!");
        chdir($ld) or expire($conn, "local cd $ld failed: $!");
        if(-r $fn) {
            my %nstrings = %$strings;
            my %npats = %$pats;
            readctl($conn, $fn, 0, \%nstrings, \%npats);
            sync($conn, "$name$ld/", $fn, \%nstrings, \%npats);
        } else {
            sync($conn, "$name$ld/", $fn, $strings, $pats);
        }
        $conn->cdup() or expire($conn, "cd up failed: !!");
        chdir('..') or expire($conn, "local cd .. failed: $!");

        delete $rdirs->{$ld};
    }

    # Wipe remotes not matched locally.
    foreach my $rd(keys %$rdirs) {
        next if($pats->{"presdir"} && ($rd =~ /$pats->{"presdir"}/));
        $conn->cwd($rd) or expire($conn, "cd $name$rd failed: !!");
        delrmt($conn, "$name$rd/");
        $conn->cdup() or expire($conn, "cd up failed: !!");
        print "Removing directory $name$rd.\n";
        if(!$norun) {
            $conn->rmdir($rd) or
                expire($conn, "Removal of $name$rd failed: !!", 0);
        }
    }
}

# Print the file times map.
sub prfl {
    my $hr = shift @_;

    foreach my $k(sort keys %$hr) {
        my $mod = localtime($hr->{$k});
        print "$k->$mod\n";
    }
}

# Synchronize clocks.
sub clocksync {
    my $conn = shift @_;
    my $fn = shift @_;

    if(! -f $fn) {
        open(SF, ">$fn") or
            expire($conn, "Cannot create $fn for time sync option");
        close(SF);
    }
    -z $fn or
        expire($conn, "File $fn for time sync must be empty.");

    $conn->put($fn) or
        expire($conn, "$fn send failed: !!");

    my $now_here = time();
    my $now_there = $conn->mdtm($fn) or
```

```
        expire($conn, "Cannot get $fn write time");

    #print "FRED: $now_here $now_there\n";
    #print "FRED: ", localtime($now_here), " ", localtime($now_there), "\n";

    $syncoff = $now_here - $now_there;
    $syncoff -= 5; # Be a bit conservative.

    #print "A: [$now_here] [$now_there] [$syncoff]\n";

    $conn->delete($fn);

    my $hrs = int($syncoff/3600);
    my $mins = int($syncoff/60) - $hrs*60;
    my $secs = $syncoff - $hrs*3600 - $mins*60;
    printf("Clock sync offset: %d:%02d:%02d\n", $hrs, $mins, $secs);
}

# Check for login info.
$strings{"host"} && $strings{"acct"} && $strings{"pwd"} or
    die "Hostname, account name, and password must be specified.\n";

# Do the communications.
my $conn = Net::FTP->new($strings{"host"}, Passive => 1) or
    die "Connect: $@\n";
$conn->login($strings{"acct"}, $strings{"pwd"}) or
    expire($conn, "Login as $strings{'acct'} failed: !!");
if($strings{"cd"}) {
    $conn->cwd($strings{"cd"}) or
        expire($conn, "Initial directory change failed: !!");
}
$conn->binary() or
    expire($conn, "Binary mode failed: !!");

# Optional clock synchronization step.
if($strings{"sync"}) { clocksync($conn, $strings{"sync"}); }

#my ($a, $b) = rmt_contents($conn);
#print "files:\n";
#prfl($a);
#print "dirs:\n";
#prfl($b);

# Now the real work.
sync($conn, '', $rcfile, \%strings, \%pats);

$conn->quit();

exit $was_err;
```

```
use strict;
use Tk;
use Tk::Dialog;
use Net::FTP;

# Colors
my @PBG = (-background => '#E6E6FA');
my @ABG = (-activebackground => '#FFE6FA');
my @PFG = (-foreground => '#0000ff');
my @AFG = (-activeforeground => '#0000ff');
my @PCL = (@PBG, @PFG);
my @BG = (@PBG, @ABG);
my @FG = (@PFG, @AFG);
my @CL = (@BG, @FG);

# Description (font and color) of a title.
my @titdesc = (-font => ['Arial', 16, 'bold'], -foreground => '#228800');

my $main = new MainWindow(@PCL);
$main->title("FTP Download");

# Login information.
my ($host, $acct, $password);

# FTP connection.
my $conn;

# Boom msg.
sub ebox {
    my $msg = shift @_;
    $msg = $msg->message() if ref $msg;

    $main->messageBox(-type => 'OK', -icon => 'error', -message => $msg);
    return;
}

# Generate s label/entry pair for the login window.  These will be
# appropriately gridded on row $row inside $par.  Text box has width
# $width and places its contents into the reference $ref.  If $ispwd,
# treat it as a password entry box.
sub genpair {
    my ($par, $row, $text, $ref, $width, $ispwd) = @_;

    my $tbut = $par->Label( -text => $text, @PCL);
    my $lab = $par->Entry(-background => 'white',
                          # -activebackground => 'white',
                          -foreground => 'black',
                          #-activeforeground => 'black',
                          -textvariable => $ref,
                          -width => $width);
    $lab->configure(-show => '*') if $ispwd;
    $tbut->grid(-row => $row, -column => 0, -sticky => 'nse');
    $lab->grid(-row => $row, -column => 1, -sticky => 'nsw');
}

# Terminate pgm.
sub term {
    $conn->quit if $conn;
    exit 0;
}

# Build the login window.
sub logscreen {
    my $parent = shift @_;

    my $row = 0;
```

```
    my $toplab = $parent->Label(-text => "FTP Server Login",
                                -justify => 'center',
                                @titdesc, @PBG);
    $toplab->grid(-row => $row++, -column => 0,
                  -columnspan => 2, -sticky => 'news');
    genpair($parent, $row++, 'Host:', \$host, 25);
    my $bframe = $parent->Frame();
    $bframe->grid(-row => $row++, -column => 0,
                  -columnspan => 2, -sticky => 'news');
    my $go = $bframe->Button(-text => 'Anon. Login',
                             -command => [ \&do_login, 0, $parent, 1 ],
                             @CL);
    $go->pack(-side => 'left', -expand => 'left', -fill => 'both');
    my $go = $bframe->Button(-text => 'User Login',
                             -command => [ \&do_login, 0, $parent, 2 ],
                             @CL);
    $go->pack(-side => 'left', -expand => 'left', -fill => 'both');

    genpair($parent, $row++, 'Login:', \$acct, 15);
    genpair($parent, $row++, 'Password:', \$password, 15, 1);


    my $stop = $parent->Button(-text => 'Exit',
                               -command => \&term , @CL);
    $stop->grid(-row => $row++, -column => 0, -columnspan => 2,
                -sticky => 'news');

    # CR same as pushing login.
    $parent->bind('<KeyPress-Return>', [ \&do_login, $parent, 3 ] );
}

# Log into the remote host.  If successful, start the directory loader.
# Modes are: 1: Anonymous, 2: User, 3: Return, which does anon if the
# user infor was not filled in, and user otw.
sub do_login {
    my ($discard, $par, $mode) = @_;

    # Adjust user data by mode.
    if($mode == 1 || ($mode == 3 && !$acct && !$password)) {
        $acct = 'anonymous';
        $password = 'anonymous' unless $password;
    }

    # Make sure we're all filled in.
    if(!$host || !$acct || !$password) {
        ebox("You must provide a host, user name and password.");
        return;
    }

    # Attempt to connect to the remote host.
    $conn = Net::FTP->new($host, Passive => 1, -debug => 1);
    if(!defined $conn) {
        ebox("FTP Connection to $host failed ($!).");
        return;
    }

    # Try the login.
    if(!$conn->login($acct, $password)) {
        ebox($conn);
        $conn->close();
        return;
    }
    $conn->binary();

    &load_dir();

    $par->destroy();

    #$conn->quit();
```

```
}

# Create the main list box with scrollbar.
my $listarea;   # Where stuff goes.
my $statuslab;  # Status information is posted here.
sub main_list {
    my $par = shift @_;

    # Label at top.
    my $toplab = $par->Label(-text => "FTP Download Agent",
                             -justify => 'center', @titdesc, @PBG);
    $toplab->pack(-side => 'top', -fill => 'x');

    # Status label.
    $statuslab = $par->Label(-text => "Not Logged In",
                             -justify => 'center', @PCL);
    $statuslab->pack(-side => 'top', -fill => 'x');

    # Exit button
    my $exbut = $par->Button(-text => "Exit", -command => \&term, @CL);
    $exbut->pack(-side => 'bottom', -fill => 'x');

    # List area with scroll bar.  The list area is disabled since we
    # don't want the user to type into it.
    $listarea = $par->Text(-height => 10, -width => 40,
                           -cursor => 'sb_left_arrow',
                           -state => 'disabled', @PCL);
    my $scr = $par->Scrollbar(-command => [ $listarea => 'yview' ], @PBG);
    $listarea->configure(-yscrollcommand => [ $scr => 'set' ]);
    $scr->pack(-side => 'right', -fill => 'y');
    $listarea->pack(-side => 'left');

    # Bind the system exit button to our exit.
    $main->protocol('WM_DELETE_WINDOW', \&term);
}

# Change the color of a tag for entering and leaving.  Unfortunately, there
# is no active color for tags in a text box.
sub recolor {
    my ($tw, $tag, $color) = @_;
    $tw->tagConfigure($tag, -foreground => $color);
    #print "FRED: $tw $tag $color\n";
}

# Do a CD and load the contents.  If there is no directory name, skip
# the CD.
sub load_dir
{
    my ($wid, $dir) = @_;

    #print "load_dir($dir)\n";

    # Change directory.
    if(@_) {
        if(!$conn->cwd($dir)) {
            ebox($conn);
            return;
        }
        $statuslab->configure(-text => "[Loading $dir]");
    } else {
        $statuslab->configure(-text => '[Loading Home Dir]');
    }
    $main->update();


    # Get the list of files.
    my @names = $conn->dir();
    if(!$conn->ok()) {
        ebox($conn);
```

```
        return;
    }

    my @files = ();
    my @dirs = ();
    my $sawdots = 0;
    while(my $n = shift @names) {
        # Split the lines (assume Unix format)
        chomp $n;

        # Real lines start with the perm bits.  And we don't want specials.
        next if $n !~ /^[\-d]([r\-][w\-][x\-]){3}/;

        # Extract the useful parts, toss the bones.
        my @parts = split(/\s+/, $n, 9);
        next if @parts < 9;
        my $fn = pop @parts;
        $sawdots = 1 if $fn eq '..';
        my $modes = shift @parts;
        if($modes =~ /^d/) {
            push @dirs, $fn;
        } else {
            push @files, $fn;
        }
    }

    # Add .. if not present, then sort the list.
    push @dirs, '..' unless $sawdots;
    @files = sort @files;
    @dirs = sort @dirs;

    # Fill in the text box.  We also bind lots of events to the file names
    # to make stuff happen when we move the mouse around.
    $listarea->configure(-state => 'normal');
    $listarea->delete('1.0','end');
    my $ct = 0;
    while(my $f = shift @dirs) {
        #print "Inserting dir: [$f]\n";
        $listarea->insert('end', "$f\n", "fn$ct");
        $listarea->tagConfigure("fn$ct", -foreground => '#4444ff');
        $listarea->tagBind("fn$ct", '<Button-1>', [ \&load_dir, $f ]);
        $listarea->tagBind("fn$ct", '<Enter>',
                           [ \&recolor, "fn$ct", '#000088' ]);
        $listarea->tagBind("fn$ct", '<Leave>',
                           [ \&recolor, "fn$ct", '#4444ff' ]);
        ++$ct;
    }

    while(my $f = shift @files) {
        #print "Inserting file: [$f]\n";
        $listarea->insert('end', "$f\n", "fn$ct");
        $listarea->tagConfigure("fn$ct", -foreground => 'red');
        $listarea->tagBind("fn$ct", '<Button-1>', [ \&dld_file, $f ]);
        $listarea->tagBind("fn$ct", '<Enter>',
                           [ \&recolor, "fn$ct", '#880000' ]);
        $listarea->tagBind("fn$ct", '<Leave>', [ \&recolor, "fn$ct", 'red' ]);
        ++$ct;
    }
    $listarea->configure(-state => 'disabled');

    # Update the status label.
    my $loc = $conn->pwd();
    if(!$loc) {
        ebox($conn);
        $statuslab->configure(-text => '???');
    } else {
        $statuslab->configure(-text => $loc);
    }
}
```

```
# Downoad the file.
sub dld_file
{
    my ($wid, $fn) = @_;

    # Announce.
    $statuslab->configure(-text => "[Retrieving $fn]");
    $main->update();

    # Get the file.
    if(!$conn->get($fn)) {
        ebox($conn);
        return;
    }

    $statuslab->configure(-text => "Got $fn");
}

#logscreen($main);

# Create the main list with the list of files.
main_list($main);

# Demand a login.
my $logwin = $main->Toplevel(@PCL);
$logwin->title("FTP Login");
logscreen($logwin);

MainLoop;
```

Створення парентів для активації клієнтів

```
$size = 0;          # 0 => line-oriented input, else fixed format.
@files = ();        # List of files
$open_new_file = 1; # force get_next_line() to open the first file
$debugging = 0;     # Enable with -d commmand line flag
$col = "";
$code = "";
generate_part1();
generate_part2();
generate_part3();
col();           # sub col has now been generated. Call it !
exit(0);


sub generate_part1 {
    # Generate the initial invariant code of sub col()
    $code  = 'sub col { my $tmp;';           # Note the single quotes
    $code .= 'while (1) {$s = get_next_line(); $col = "";';
    $delimiter = '|';
}

sub generate_part2 {
    # Process arguments
    my ($col1, $col2);
    foreach $arg (@ARGV) {
        if (($col1, $col2) = ($arg =~ /^(\d+)-(\d+)/)) {
            $col1--;# Make it 0 based
            $offset = $col2 - $col1;
            add_range($col1, $offset);
        } elsif (($col1, $offset) = ($arg =~ /^(\d+)\+(\d+)/)) {
            $col1--;
            add_range($col1, $offset);
        } elsif ($size = ($arg =~ /-s(\d+)/)) {
            # noop
        } elsif ($arg =~ /^-d/) {
            $debugging = 1;
        } else {
            # Must be a file name
            push (@files, $arg);
        }
    }
}

sub generate_part3 {
    $code .= 'print $col, "\n";}}';

    print $code if $debugging; # -d flag enables debugging.
    eval $code;
    if ($@) {
        die "Error ..........\n $@\n $code \n";
    }
}

sub add_range {
    my ($col1, $numChars) = @_;
    # substr complains (under -w) if we look past the end of a string
    # To prevent this, pad the string with spaces if necessary.
    $code .= "\$s .= ' ' x ($col1 + $numChars - length(\$s))";
    $code .= "    if (length(\$s) < ($col1+$numChars)
);";
    $code .= "\$tmp = substr(\$s, $col1, $numChars);";
    $code .= '$tmp .= " " x (' . $numChars .  ' - length($tmp));';
    $code .= "\$col .= '$delimiter' . \$tmp; ";
}

sub get_next_line {
    my($buf);
```

```
  NEXTFILE:
    if ($open_new_file) {
        $file = shift @files || exit(0);
        open (F, $file) || die "$@ \n";
        $open_new_file = 0;
    }
    if ($size) {
        read(F, $buf, $size);
    } else {
        $buf = <F>;
    }
    if (! $buf) {
        close(F);
        $open_new_file = 1;
        goto NEXTFILE;
    }
    chomp($buf);
    # Convert tabs to spaces (assume tab stop width == 8)

    # expand leading tabs first--the common case
    $buf =~ s/^(\t+)/' ' x (8 * length($1))/e;

    # Now look for nested tabs. Have to expand them one at a time - hence
    # the while loop. In each iteration, a tab is replaced by the number of
    # spaces left till the next tab-stop. The loop exits when there are
    # no more tabs left
    1 while ($buf =~ s/\t/' ' x (8 - length($`)%8)/e);

    $buf;
}
```

Створення активації та авторизації клієнтів

```
use Tk;

print STDERR "Scouting man directories\n";
scout_man_dirs();
print STDERR "Starting UI ...";
create_ui();
print STDERR "Done \n";
MainLoop();
exit(0);

#-----------------------------------------------------------------------
my $menu_headings;        # "Headings" MenuButton
my $ignore_case;          # 1 if check-button on in Search menu
my $match_type;           # '-regexp' or '-exact'.
my $text;                 # Main text widget
my $show;                 # "Show" entry widget
my $search;               # "Search" entry widget
my %sections;             # Maps section ('1', '3' ,'3n' etc.)
                          #  to list of topics in that section

sub show_man {
    my $entry = $show->get();   # get entry from $show
    my ($man, $section) = ($entry =~ /^(\w+)(\(.*\))?/);
    if ($section && (!is_valid_section($section))) {
        undef $section ;
    }
    my $cmd_line = get_command_line($man, $section); # used by open

    # Erase everything to do with current page (contents, menus, marks)
    $text->delete('1.0', 'end');  # erase current page
    $text->insert('end', "Formatting \"$man\" .. please wait", 'section');
    $text->update();                  # Flush changes to text widget
    $menu_headings->menu()->delete(0,'end'); # Delete current headings
    my $mark;
    foreach $mark ($text->markNames) {  # remove all marks
        $text->markUnset($mark);
    }

    # UI is clean now. Open the file
    if (!open (F, $cmd_line)) {
        # Use the text widget for error messages
        $text->insert('end', "\nError in running man or rman");
        $text->update();
        return;
    }
    # Erase the "Formatting $man ..." message
    $text->delete('1.0', 'end');
    my $lines_added = 0; my $line;

    while ($line = <F>) {
        $lines_added = 1;
        # If first character is a capital letter, it's likely a section
        if ($line =~ /^[A-Z]/) {
            # Likely a section heading
            ($mark = $line) =~ s/\s.*$//g;  # $mark has section title
            my $index = $text->index('end');# note current end location
            # Give 'section' tag to the section title
            $text->insert('end', "$mark\n\n", 'section');
            # Create a menu entry. Have callback invoke text widget's
            # 'see' method to go to the index noted above
            $menu_headings->command(
                    '-label' => $mark,
                    '-command' => [sub {$text->see($_[0])},$index])
        } else {
            $text->insert('end', $line); # Ordinary text. Just insert.
        }
    }
    if ( ! $lines_added ) {
```

```
        $text->insert('end', "Sorry. No information found on $man");
    }
    close(F);
}

sub get_command_line {
    my ($man, $section) = @_; # Given topic and section, construct
                              # Unix command-line
    if ($section) {
        $section =~ s/[()]//g; # remove parens
        return "man -s $section $man 2> /dev/null | rman |";
    } else {
        return "man $man 2> /dev/null | rman |";
    }
}

sub create_ui {
    my $top = MainWindow->new();

    # MENU STUFF

    # Menu bar
    my $menu_bar = $top->Frame()->pack('-side' => 'top', '-fill' => 'x');

    # File menu
    my $menu_file = $menu_bar->Menubutton('-text' => 'File',
                                          '-relief' => 'raised',
                                          '-borderwidth' => 2,
                                          )->pack('-side' => 'left',
                                                  '-padx' => 2,
                                                 );
    $menu_file->separator();
    $menu_file->command('-label' => 'Quit', '-command' => sub {exit(0)});

    #Sections Menu
    $menu_headings = $menu_bar->Menubutton('-text' => 'Headings',
                                           '-relief' => 'raised',
                                           '-borderwidth' => 2,
                                           )->pack('-side' => 'left',
                                                   '-padx' => 2,
                                                   );
    $menu_headings->separator();


    #Search menu
    my $search_mb = $menu_bar->Menubutton('-text'        => 'Search',
                                          '-relief'      => 'raised',
                                          '-borderwidth' => 2,
                                          )->pack('-side' => 'left',
                                                  '-padx' => 2
                                                );
    $match_type = "-regexp"; $ignore_case = 1;
    $search_mb->separator();

    # Regexp match
    $search_mb->radiobutton('-label'    => 'Regexp match',
                            '-value'    => '-regexp',
                            '-variable' => \$match_type);
    # Exact match
    $search_mb->radiobutton('-label'    => 'Exact match',
                            '-value'    => '-exact',
                            '-variable' => \$match_type);
    $search_mb->separator();
    # Ignore case
    $search_mb->checkbutton('-label'    => 'Ignore case?',
                            '-variable' => \$ignore_case);


    #Sections Menu
```

```
    my $menu_sections = $menu_bar->Menubutton('-text' => 'Sections',
                                              '-relief' => 'raised',
                                              '-borderwidth' => 2,
                                              )->pack('-side' => 'left',
                                                      '-padx' => 2,
                                                      );
    # Populate sections menu with keys of % sections
    my $section_name;
    foreach $section_name (sort keys %sections) {
        $menu_sections->command (
                 '-label' => "($section_name)",
                 '-command' => [\&show_section_contents, $section_name]);
    }

    # TEXT STUFF

    $text = $top->Text ('-width' =>  80,
                        '-height' => 40)->pack();
    $text->tagConfigure('section',
                        '-font' => '-adobe-helvetica-bold-r-normal--14-140-75-
75-p-82-iso8859-1');
    $text->bind('<Double-1>', \&pick_word);
    $top->Label('-text' => 'Show:')->pack('-side' => 'left');

    $show = $top->Entry ('-width'   =>  20,
                         )->pack('-side' => 'left');
    $show->bind('<KeyPress-Return>', \&show_man);

    $top->Label('-text' => 'Search:'
                )->pack('-side' => 'left', '-padx' => 10);
    $search = $top->Entry ('-width' => 20,
                           )->pack('-side' => 'left');
    $search->bind('<KeyPress-Return>', \&search);
}

sub is_valid_section {
    my $section= $_[0];
    return 0 unless $section =~ /\((.*?)\)/;
    my $section = $1;
    my $s;
    foreach $s (keys %sections) {
        if (lc($s) eq lc($section)) {
            return 1;
        }
    }
    0;
}

sub pick_word {
    my $start_index = $text->index('insert wordstart');
    my $end_index = $text->index('insert lineend');
    my $line = $text->get($start_index, $end_index);
    my ($page, $section) = ($line =~ /^(\w+)(\(.*?\))?/);
    return unless $page;
    $show->delete('0', 'end');
    if ($section && is_valid_section($section)) {
        $show->insert('end', "$page${section}");
    } else {
        $show->insert('end', $page);
    }
    show_man();
}

sub show_section_contents {
    my $current_section = $_[0];
    $text->delete('1.0', 'end');
    $menu_headings->menu()->delete(0,'end');
    my ($i, $len);
    return unless exists $sections{$current_section};
```

```
    my $spaces = " " x 40;
    my $words_in_line = 0;  # New line when this goes to three
    my $man;
    foreach $man (@{$sections{$current_section}}) {
        $text->insert('end', $man . substr($spaces,0, 24 - length($man)));
        if (++$words_in_line  == 3) {
            $text->insert('end', "\n");
            $words_in_line = 0;
        }
    }
}

sub search {
    my $search_pattern = $search->get();
    $text->tagDelete('search');
    $text->tagConfigure('search',
                        '-background' => 'yellow',
                        '-foreground' => 'red');

    my $current = '1.0'; my $length = '0';
    while (1) {
        if ($ignore_case) {
            $current = $text->search('-count' => \$length,
                                     $match_type,
                                     '-nocase',
                                     '--',
                                     $search_pattern,
                                     $current,
                                     'end');
        } else {
            $current = $text->search('-count' => \$length,
                                     $match_type,
                                     '--',
                                     $search_pattern,
                                     $current,
                                     'end');
        }
        last if (!$current);
        $text->tagAdd('search', $current, "$current + $length char");
        $current = $text->index("$current + $length char");
    }
}


use Cwd;
sub scout_man_dirs {
    my (@man_dirs,$man_dir, $section);
    if ($ENV{MANPATH}) {
        @man_dirs = split (/:/, $ENV{MANPATH});
    } else {
        push (@man_dirs, "/usr/man");
    }
    # Convert all relative man paths to fully qualified ones, by
    # prepending with $cwd
    my $cwd = cwd();
    foreach $man_dir (@man_dirs) {
        next if ($man_dir =~ m|^/|);
        $man_dir = "$cwd/$man_dir"; # Modifies entry in man_dirs
    }
    foreach $man_dir (@man_dirs) {
        chdir $man_dir || next;
        # Now, in /usr/man, say. Get all the directories
        my @section_dirs = grep {-d $_} <man*>;
        my $section_dir;
        # @section_dirs has man1, man2, man3s etc.
        foreach $section_dir (@section_dirs) {
            chdir $section_dir || next;
            ($section = $section_dir) =~ s/^man//;
            push (@{$sections{$section}}, <*.$section>);
```

```
            chdir "..";
        }
        chdir "..";
    }
    # All sections in all man pages have been slurped in. Remove duplicates
    foreach $section (keys %sections) {
        my @new_list;
        my %seen;
        @new_list = sort (grep (!$seen{$_}++, @{$sections{$section}}));
        # Change all entries like cc.1 to cc(1)
        foreach (@new_list) {
            $_ =~ s/[.](.*)/($section)/;
        }
        $sections{$section} = \@new_list;
    }
}
```