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 віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів”

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

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 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
« 6 » вересня 2024 року

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

Білану Ярославу Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів*

2. Керівник роботи *Петренюк Володимир Ілліч, 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 *2.12.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів*

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-проєкту.*
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., к.т.н., доцент	06.10.2024	16.11.2024

7. Дата видачі завдання « 6 » вересня 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	10.10.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2024 р.	
3.	Розробка моделі компонента	20.10.2024 р.	
4.	Розробка структур даних	25.10.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2024 р.	
6.	Програмування алгоритмів	10.11.2024 р.	
7.	Розрахунок економічної ефективності	13.11.2024 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2024 р.	
9.	Оформлення ПЗ	17.11.2024 р.	
10.	Попередній захист роботи	2.12.2024 р.	

Дата видачі завдання
« 6 » вересня 2024 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2024 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Білан Я.Д. Дослідження та програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Метою розробки є дослідження та програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Об'єктом дослідження є процес віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Предметом дослідження є методи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, адміністрування, сервер, KVM-over-IP-перемикачів

ABSTRACT

Bilan Ya.D. Research and software implementation of a remote server administration system using KVM-over-IP switches. 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, which is intended for the system of remote administration of servers using KVM-over-IP switches.

The purpose of the development is the research and software implementation of the remote server administration system using KVM-over-IP switches.

The object of research is the process of remote server administration using KVM-over-IP switches.

The subject of research is methods of remote server administration using KVM-over-IP switches.

Research methods are based on computer network theory methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the remote server administration system using KVM-over-IP switches.

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

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

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

The program was developed in the Python environment.

Keywords: computer engineering, administration, server, KVM-over-IP switches

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 7
1.1 Призначення системи..... 7
1.2 Область застосування..... 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 12
2.3 Розгорнута постановка завдання 13
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 15
3.1 Опис функціонування системи 15
3.2 Розробка структурної схеми..... 19
3.3 Розробка функціональної схеми 29
3.4 Розробка діаграми процесів..... 39
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 41
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 41
4.2 Захист розробленого програмного забезпечення..... 49
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 54
6 НАУКОВА НОВИЗНА 58

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ІТ-ПРОЄКТУ 59
7.1 Визначення цільової аудиторії кінцевого готового продукту 59
7.2 Оцінка привабливості шляхом застосування методів експертних оцінок ... 60
7.3 Вибір методу оцінки вартості ПЗ 61
7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості..... 62
7.5 Пропозиція алгоритму просування проєкту розробки ПЗ 64
7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ 64
7.7 Визначення ключових факторів успіху конкретного проєкту..... 66
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 68
8.1 Аналіз умов праці на робочому місці програміста 68
8.2 Розробка заходів з умов поліпшення охорони праці 71
8.3 Дослідження інформаційного навантаження на програміста..... 73
8.4 Висновки до розділу..... 76
9 ОСНОВНІ ВИСНОВКИ..... 77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕОМ	–	електрона обчислювальна машина
НСД	–	несанкціонований доступ
ПЗ	–	програмне забезпечення
СУБД	–	система управління базами даних
СВВ	–	система виявлення вторгнень

ВСТУП

Актуальність теми. Для рішення завдань керування, економії простору в серверних шафах і рятування від зайвих кабелів багато адміністраторів використовують перемикачі клавіатури, відео й миші (Keyboard, Video, Mouse, KVM). Ці пристрої забезпечують контроль за безліччю серверів і іншим устаткуванням з однієї консолі, причому сервери зовсім не повинні бути однотипними: вони можуть працювати під керуванням самих різних систем – Windows, Unix, Linux.

Щоб ближче познайомитися з можливостями сучасних KVM, їхньою роллю й місцем у керуванні сучасної серверної й IT-інфраструктурою, у роботі розробляється програмне забезпечення, що дозволяє реалізувати потребу в модернізації серверного парку. Використовувані в її ЦОД перемикачі KVM застаріли – зокрема, вони можуть підтримувати тільки обмежене число серверів і не забезпечують віддалений доступ.

Отже, компанія збирається модернізувати свій серверний парк у ЦОД, а заодно обновити й уніфікувати керування серверами. На площадці розміщається 20 стійок з устаткуванням, половину з них займають успадковані різнотипні сервери різних форм-факторів і конфігурацій. Заповнення стійок нерівномірне – від декількох серверів, у тому числі із блейд-системами й UNIX-системами, до 20 і більше. Стійки, що залишилися, планується заповнити однотипними серверами під віртуалізацію (остаточного рішення про вибір моделі ще не прийняте).

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

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

– Огляд існуючих систем віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

– Дослідження системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

– Програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Об'єктом дослідження є процес віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Предметом дослідження є методи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

– Розроблено вітчизняний продукт віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

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

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2024 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Замовник хотів би мати контроль за всіма серверами, у тому числі віртуальними, через один інтерфейс. Через критичність керування повинне бути реалізоване по зовнішньому каналу (out-of-band). Оператори будуть розміщатися в сусідньому залі, однак згодом керування цією й іншою площадками планується зробити єдиним і здійснювати із центрального офісу, так що варто передбачити підтримку віддаленого доступу через VPN. Система доступу повинна підтримувати надійну автентифікацію й розмежування прав доступу.

Компанія хотіла б одержати відповіді на наступні питання:

– Яке рішення ви могли б порекомендувати (включаючи встаткування й ПЗ)?

– Оскільки потрібне поділ доступу й рівнів доступу, які механізми безпеки можуть застосовуватися при локальному й віддаленому доступі?

– Яким образом можна інтегрувати в єдину систему консольні KVM, розташовані в декількох стійках (чи можливо це в принципі, від чого залежить можливість їхнього подальшого використання)?

– Які вимоги до каналу для віддаленого керування при максимальному дозволі, яке дозвіл забезпечується при підключенні по мобільному зв'язку, до скількох серверів може одночасно здійснюватися доступ без зниження якості роботи віддалених операторів і в цілому які відмінності цього підходу від локального доступу?

1.2 Область застосування

Сучасному ЦОД необхідні ефективні засоби для контролю за безліччю серверів, як фізичних, так і віртуальних, при цьому одним з інструментів такого контролю є KVM. Головна ж «об'єднуюча» роль для всіх засобів віддаленого доступу належить системі централізованого керування. Загальним для всіх пропозицій стало використання цифрових перемикачів IP KVM.

Для централізації всієї інформації й організації єдиної точки входу для керування використовується програмний пакет. Він установлюється на сервер і надає доступ через браузер до будь-якої функції по зовнішньому каналу керування з будь-якої точки мережі, навіть якщо вона перебуває на іншому континенті. У самому UMG є автономний Web-інтерфейс, а для роботи в приміщенні ЦОД – локальний набір інтерфейсів, до яких можна підключити LCD-консоль. Програмний пакет дозволяє використовувати всі функції сервісних процесорів: включення/вимикання блоків живлення, відстеження параметрів фізичного середовища й навіть одержання доступу до робочого стола сервера (віртуальний KVM). Крім того, інтегруючись на програмному рівні з VMware, Citrix і Microsoft HV, це ПЗ дає можливість віддаленно працювати з віртуальними серверами так само, як з фізичними.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Aten і «Колан»

Aten і «Колан» обмежилися рекомендацією використовувати перемикачі серії KN (зокрема, IP KVM KN4124V) і програмного комплексу для централізованого керування CC2000. У кожній стійці пропонується встановити по одному IP KVM KN4124V, що дозволяє управляти 24 серверами. При цьому для одних стійок така кількість керованих серверів може виявитися надмірною, а для інших – недостатньою. Втім, в Aten у тій же серії IP KVM є пристрої з підтримкою 8, 16, 32 і 40 серверів. У свою чергу, CC2000 дозволяє централізовано управляти 10 перемикачами серії KN, тобто контролювати сервери в 10 стійках. У принципі, CC2000 може управляти необмеженою кількістю KVM і ліцензується виходячи з наявної кількості серверів понодно (НОД – це сума портів KVM (один), rs232 (один) і керування електроживленням (два)). Для керування через послідовні порти застосовуються пристрої серії SN з 1/8/16/32/48 портами, а для управління електроживлення – пристрою серії PE з індексом R.

Нієншанц-автоматика

Як рішення по уніфікованому керуванню серверами компанія «Нієншанц-автоматика» пропонує комплекс із over-IP-перемикачів Dominion KXIII виробництва Raritan і її ж системи централізованого керування CommandCenter SecureGateway. Для доступу до кожної окремої стійки (або парі стійкий, залежно від розміщення) досить одного перемикача KVM-over-IP з відповідною кількістю інтерфейсних CIM-модулів, що підключаються до KVM-портів кожного сервера.

В Raritan CIM-модулі є не тільки для VGA і DVI, але й для HDMI і DisplayPort. Таким чином, KVM-перемикачі Raritan сумісні не тільки з існуючою, у тому числі й застарілому обладнанні, але й з майбутніми моделями. Керування серверами може здійснюватися централізовано в межах мережі IP з будь-якого зручного місця (у тому числі й через VPN-з'єднання). При необхідності можна одержати локальний доступ, підключивши консоль безпосередньо до KVM-перемикача. Користувальницький інтерфейс при локальному й віддаленому доступі ідентичний, написаний на HTML і не вимагає установки спеціального ПЗ.

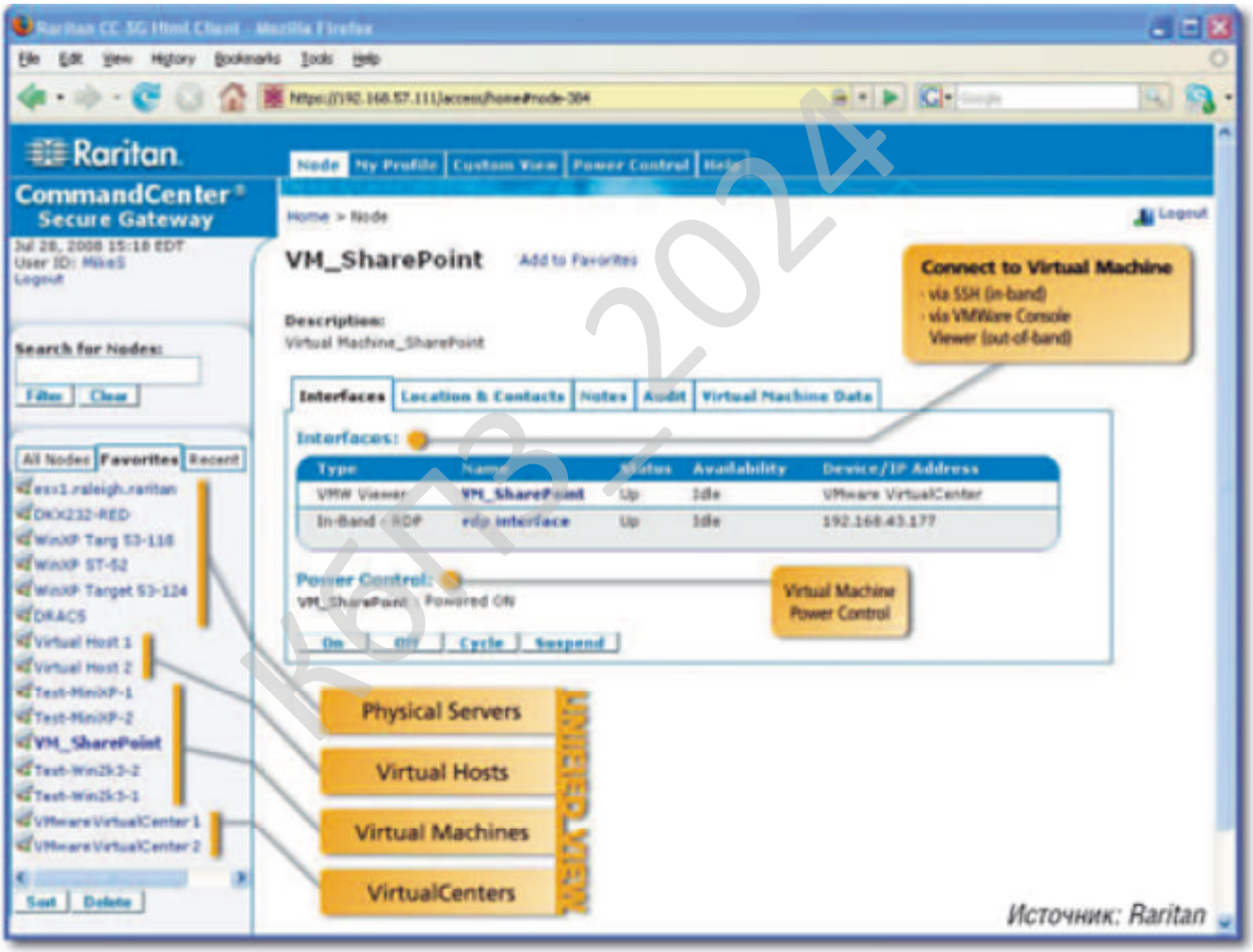

Рисунок 2.1 – Система централізованого керування Raritan CommandCenter SecureGateway надає «єдину точку входу», дозволяючи управляти всім спектром наявного встаткування, у тому числі віртуальними пристроями.

Avocent

Avocent запропонувала свій відносно новий пристрій (так званий Unified Management Gateway – UMG), що, як уважають у компанії, буде особливо корисно при тім серверному «зоопарку», який існує в ЦОД замовника, тим більше що немає визначеності щодо вибору між KVM і серверними процесорами. UMG поєднує в собі всю функціональність віддаленого (out-of-band) керування: до кожного з 40 портів UMG можна приєднати KVM-порт сервера (де є VGA і USB), порт RS-232 або серверний процесор (IPMI, HP iLo, IBM RSA, Dell DRAC і т.д.). При цьому UMG автоматично розпізнає, який саме інтерфейс до нього підключений.

DVI KVM-подовжувач із доступом через IP

DVI KVM over IP подовжувач Altusen KE6900/KE6940 компанії Aten дозволяє одержати доступ до комп'ютерної системи з віддаленої USB-консолі, що підтримує USB-клавіатуру, USB-мишу й монітор з інтерфейсом DVI через закриту мережу Intranet. KE6900 Single View Extender підтримує один дисплей з інтерфейсом DVI з кожної сторони, а KE6940 Dual View Extender – два дисплеї з інтерфейсом DVI з кожної сторони, забезпечуючи вивід відео на два монітори. Як матричний подовжувач пристрій здатний підтримувати одиничні й множинні підключення комп'ютерів до консолей: один до одному (у режимі подовжувача), один до багатьох (у режимі поділу), багато хто до одному (у режимі перемикача) або багато до багатьох (у матричному режимі).

Рисунок 2.2 – DVI KVM over IP подовжувач Altusen KE6900/KE6940

Серія KE6900/KE6940 дозволяє одержати доступ до комп'ютерів у локальній мережі й управляти їхніми ресурсами за допомогою будь-який KVM-консолі, також підключеної до мережі. Подовжувач складається із блоку передавача, що з'єднується з комп'ютером, і блоку приймача, що забезпечує консольний доступ. Консоль може підключатися до віддаленого комп'ютера прямо за допомогою стандартного кабелю Категорії 5e і дозволяє виконувати адміністрування за принципом «точка – точка», «точка – безліч точок» або «безліч точок – безліч точок».

Програмне забезпечення KE Matrix Manager надає розширені можливості для автентифікації по обліковому записі користувача й паролю, а також для автоматичного визначення всіх пристроїв KE6900/KE6940 у тієї ж підмережі й виявлення з'єднань, які можуть бути підключені й надані для спільного доступу. Нові функції безпеки забезпечують додатковий захист і підтримку 128-розрядного шифрування AES для передачі важливої інформації. Ще один рівень безпеки з'єднання реалізований завдяки застосуванню RADIUS, LDAP, AD і автентифікації віддаленого користувача.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також

містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Одне з основних питань, що очікуване хвилює замовника, в тому що стосується доступу, особливо віддаленого, до критичного встаткування, – це безпека. Надійність і безпека – от основні критерії, якими необхідно керуватися при виборі рішення для централізованого керування засобами віддаленого доступу. Швидше за все, така стисліть пояснюється тим, що постачальники перемикачів KVM традиційно приділяють підвищену увагу темі захисту й розмежування доступу, тому наявність відповідних розвинених засобів сприймається як щось саме собою що розуміє, не потребуючих подальших пояснень.

Щоб одержувати доступ до конкретних комп'ютерів і управляти ними могли тільки авторизовані користувачі, у перемикачах Aten підтримується дворівневий парольний захист. У перемикачах застосовуються сильні алгоритми шифрування, DES, 3DES, AES; KVM, відео й інформація з віддаленого носія шифруються окремо. Канали можуть шифруватися різними алгоритмами, у тому числі й з довільним вибором на один сеанс. У свою чергу, система централізованого керування на базі CC2000 і перемикачів IP KVM підтримує багаторівневий доступ і дозволяє призначати наступні стандартні ролі: суперадміністратор, адміністратор, користувач, аудитор. При необхідності можна вводити власні ролі, виходячи з маски прав. Можливі зовнішня автентифікація й авторизація RADIUS, TACACS, LDAP, LDAPS, Active Directory.

У перемикачах Raritan здійснюється шифрування AES, авторизація може вироблятися за допомогою як убудованих засобів системи, так і зовнішніх серверів автентифікації LDAP/Radius. Система централізованого керування CommandCenter Secure Gateway забезпечує гнучке й зручне настроювання

політик безпеки для окремих користувачів і цілих груп, надаючи «єдину точку входу» зі зручним і інтуїтивно зрозумілим інтерфейсом для доступу до відповідних серверів.

В програмному забезпеченні передбачені механізми поділу прав доступу до об'єктів і ролі кожного користувача (які сервіси йому будуть доступні при роботі із цим ПЗ), підтримується інтеграція з різними службами каталогів, такими як AD, LDAP, TACACS і RADIUS.

Якщо виходити із цих коротких описів, всі рішення оснащені приблизно однаковими функціями безпеки. Зокрема, у мережах Windows буде корисна можливість реєстрації на перемикачі IP KVM з використанням засобів автентифікації Active Directory і наданням різних рівнів доступу до підключених пристроїв на підставі наявних профілів користувачів Active Directory.

При каскадному підключенні один перемикач KVM може управляти декількома пристроями. Звичайно досить лише KVM-консолі KL1516A, але в деяких випадках адміністратори воліють застосовувати матричний KVM-перемикач KM0932/0532, тому що ці пристрої можуть інтегруватися з перемикачами серії KN (з керуванням по мережі) і використовувати єдиний графічний інтерфейс GUI, що набагато зручніше, оскільки відпадає необхідність двічі реєструватися в системі (login) – тим самим досягається краща інтеграція.

Можна використовувати успадковані KVM, якщо об'єднати їх під керуванням централізованої системи ATEN CC2000. Консольні KVM третіх виробників у принципі можуть бути інтегровані за схемою з'єднання порт – консоль, але без інтеграції меню і єдиної авторизації, у зв'язку із чим рекомендується впроваджувати консолідоване рішення від одного виробника.

Інтеграція вже наявних KVM-перемикачів можлива на рівні інтерфейсу перемикача (наприклад, за допомогою KVM-подовжувача KXII-101-V2), тоді із централізованої системи можна одержати доступ до KVM-перемикача, а через нього – до цільових серверів. Для організації повністю «прозорого» безшовного доступу до цільових серверів (із централізованим керуванням правами

користувача) підійдуть перемикачі KVM-over-IP серій KX, консольні сервери SX, гібридні пристрої KSX, а також матричні KVM-перемикачі серії Paragon.

Програмне забезпечення дозволяє додавати в систему пристрою KVM з марками IBM, Fujitsu і Dell (вони виробляються Avocent). Повноцінна інтеграція з іншими виробниками не підтримується.

Сучасні перемикачі KVM здатні підтримувати дозвіл Full HD. Бажаючи забезпечити максимально комфортні умови для роботи системних адміністраторів, замовник вирішив поцікавитися, який канал буде потрібно для передачі зображення з максимальним дозволом. При локальному доступі, коли підключення здійснюється по окремих кабелях, це не представляє проблеми, а при віддаленому – навряд чи хтось захоче виділити такий канал лише для забезпечення зручності віддаленого керування. Якщо ж до питання віддаленого доступу підійти з іншого боку, то при твердих обмеженнях на пропускну здатність, наприклад при доступі по мобільному зв'язку, виявляється досить 128 Кбіт/с і навіть менше.

Для підтримки відео максимально можливої якості в режимі 1600?1200 у випадку перемикачів Aten для кожного підключення буде потрібно канал 50 Мбіт/с. Мінімально ж необхідна для роботи пропускна здатність становить 128 Кбіт/с, причому пристрій може гнучко підбудовуватися під неї, аж до переходу на напівтонове, чорно-біле зображення. Однак для відносно комфортного використання окремих функцій Virtual Media потрібен канал більшої ширини.

Необхідна величина пропускної здатності каналу передачі даних для сеансу KVM – це завжди компроміс між якістю й швидкістю. Завдяки використанню передових апаратно-програмних рішень для компресії даних, over-IP-перемикачі Dominion KXIII компанії Raritan дозволяють передавати відео FullHD 1080p зі звуком по каналі до 30 Мбіт/с. Але, як правило, на практиці потрібна передача набагато більше скромних обсягів даних – наприклад, при типовій роботі з текстовими документами в інтерфейсі Windows завантаження каналу становить до 100 Кбіт/с. Однак і цей рівень може бути знижений у кілька

разів за рахунок більше агресивних настроювань компресії, зменшення глибини кольору й дозволи. Таким чином, прийнятні умови для роботи можна одержати навіть при аварійному доступі через 3G-модем.

Для віддаленої роботи через програмне забезпечення не потрібно значної пропускної здатності – у мінімальному варіанті досить навіть модемного доступу. В програмному забезпеченні є механізм підстроювання під будь-яку швидкість каналу шляхом скорочення розміру вікна або кількості використовуваних квітів.

Що стосується забезпечення можливості віддаленої роботи відразу для декількох операторів, те все залежить від кількості відповідних шин/каналів у конкретного KVM. Наприклад, KVM KN 4124V с керуванням по мережі дозволяє підключити чотири віддалених канали, при цьому зниження якості не відбувається. Віддалений доступ – це так зване цифрове з'єднання, коли сигнали KVM і відео передаються в стислому виді; при локальному ж доступі звичайно використовується аналогове з'єднання, а якість передачі аналогового відео погіршується в міру збільшення відстані.

На відміну від ряду інших рішень, у випадку шлюзу UMG компанії Avocent його підключення до сервера є не аналоговим, а цифровим (TCP). Відповідно, воно краще захищено від перешкод і загасань сигналу з відстанню – для KVM-портів гарантується дальність 100 м від UMG до сервера. Для підключення KVM-портів використовуються спеціальні інтелектуальні кабельні інтерфейси, які й виконують всю оцифровку сигналів з інтерфейсних портів сервера й забезпечують їхню передачу по звичайному кабелі Категорії 5. Завдяки тому, що процесор, що обробляє всі сигнали, переміщений з KVM в інтерфейсний модуль, стало можливо одночасно відкривати до 40 вікон, що дозволяють працювати з віддаленими серверами, тоді як раніше – не більше 8.

Якщо при звертанні до конкретного перемикача обмеження на кількість одночасно працюючих операторів звичайно досить тверді, то при використанні програмного забезпечення, зокрема CC2000 компанії Aten, кількість серверів, що обслуговуються одночасно, не обмежується. Однак, якщо використовується

режим проксі, число серверів, що обслуговуються одночасно, залежить від потужності комп'ютера, на якому встановлено ПЗ, і пропускної здатності мережних інтерфейсів.

3.2 Розробка структурної схеми

Розглянемо доцільність застосування KVM при наявності в серверах серверних процесорів HP iLO, IBM RSA, Dell DRAC і ін. Тим більше що вони забезпечують ті ж (і навіть більші) можливості, що й перемикачі, – наприклад, доступ по окремому каналі, керування на рівні BIOS, підключення віртуальних носіїв (Virtual Media) і навіть включення/відключення живлення (у випадку KVM для цього потрібна інтеграція з PDU). Проте постачальники KVM не вважають, що їхньому встаткуванню не залишилося місця в серверні й ЦОД. Більше того, вони як і раніше наполягають на перевагах їхнього застосування. Аргументи наступні.

Перемикачі KVM-over-IP дозволяють одержати повний доступ до цільового сервера на рівні BIOS без необхідності придбання додаткових ліцензій або установки спеціалізованого ПЗ, що принципово відрізняє їх від програмних рішень і засобів доступу на базі сервісних процесорів. Останнім потрібно виділити окрему IP-адресу, що скорочує пул доступних адрес (для великих ЦОД це є актуальною проблемою), не говорячи про необхідність відповідних фізичних підключень. У випадку ж використання KVM досить одного IP-адреси (або двох при необхідності резервування). Крім того, сервісному процесору для роботи потрібні відкриті порти, що створює потенційну уразливість у системі.

Щоб побачити, що відбувається в момент помилки сервера, адміністраторові прийде перевірити сервер через локальну консоль: «KVM використовується для централізованого керування, для спільного використання клавіатури/миші/монітора, в остаточному підсумку – для економії сил/грошей/місця, а також як альтернативне підключення до клієнтських

серверів». До того ж у такому «зоопарку», тобто при наявності серверів різних вендорів, погоджене використання IPMI навряд чи можливо – у цьому випадку однозначно потрібні перемикачі KVM. Крім того, за допомогою перемикачів KVM серії KN можна реєструвати все, що відбувається з підключеними серверами, – для цього потрібно окремо встановити ПЗ.

Крім обґрунтування доцільності використання KVM поряд із серверними процесорами, замовника цікавила можливість контролю й за іншим установленим у стійках устаткуванням. А саме – що додатково буде потрібно для інтеграції з послідовними консолями для забезпечення керування комутаторами, маршрутизаторами й іншим устаткуванням, не оснащеним портами KVM?

Керування послідовними консолями (комутаторами, маршрутизаторами й іншим устаткуванням без портів KVM) здійснюється декількома способами, і вибір оптимального залежить від конфігурації/розподілу відповідних пристроїв по стійках. Але можна назвати три основних способи доступу:

– over-IP-перемикач KX з модулями CIM для послідовних портів;

– сервер консолей SX з підтримкою до 48 послідовних портів;

– комбінований пристрій KSX з портами KVM і послідовними портами.

Пропонується наступний варіант – придбати консольні сервери для керування послідовними пристроями. Консольний сервер забезпечує як дистанційний, так і внутримережний доступ одночасно до 48 серверів або послідовних пристроїв (у наявності є також моделі з 1, 8, 16 і 32 портами). KVM-перемикачі серії KN оснащені портами RS232, до яких можна підключити відповідне IT-устаткування, або для підключення до KVM консольного порту IT-пристрою можна скористатися модулем KA7140, що спеціально для цього призначений.

Ще один важливий момент, що замовник випустив з уваги, це перезавантаження «завислих» серверів, що KVM самі по собі не забезпечують. Для цього в сучасних моделях перемикачів передбачається інтеграція із блоками розеток (Power Distribution Union, PDU). Система пропонує й ряд додаткових

можливостей, зокрема по інтеграції із системами розподілу й керування електроживленням. Управляючи інтелектуальним пристроєм розподілу електроживлення (iPDU) на рівні окремої розетки, можна примусово включати, виключати або перезавантажувати різне встаткування. Найчастіше, при збої ПЗ, перезавантаження по живленню є єдиним реальним способом повернути працездатність тому або іншому встаткуванню.

У процесі розширення інфраструктури IT-департамент зіштовхується з необхідністю централізованого контролю за безліччю пристроїв. При ближчому розгляді з'ясовується, що потрібно організувати не тільки моніторинг наявного встаткування, але й керування ім. Обидві завдання рідко виникають окремо одна від інший, і вирішувати їх необхідно паралельно.

Як правило, на первісному етапі найбільш актуальн і пророблені є завдання керування серверами. Для повсякденної роботи (установки сервера, настроювання ПЗ й т.п.) потрібен локальний доступ, тобто KVM-консоль, що через комутатор KVM підключають відразу до декількох серверів. Для невеликих інфраструктур така схема буде прийнятної й корисної по наступних причинах:

– KVM-консоль дозволяє працювати з усіма серверами;

– поряд з функцією локального керування й перемикання між серверами, більшість KVM мають функцію віддаленого доступу – усувати збої можна з будь-якої точки миру через Інтернет;

– для роботи з подібним устаткуванням (KVM-консолі й перемикачі) не потрібні особливі знання по його установці, настроюванню й використанню.

Проте даний підхід не позбавлений істотних недоліків:

– KVM-консолі й перемикачі можна використовувати тільки при включеному й справному сервері;

– через таке підключення не можна перезавантажити завислий сервер;

– при неприступності сервера неможливо діагностувати несправність – чи поломка це самого сервера або, наприклад, проста відсутність живлення.

Нарешті, цей підхід застосуємо тільки для серверів, а на роботу IT-систем прямо впливає й інше встаткування – комутатори, системи зберігання даних і т.д., які залишаються «за бортом» подібного «централізованого» керування.

Таким чином, за допомогою перемикачів KVM можна вирішувати деяке обмежене коло щоденних завдань, а також діагностувати невеликий перелік несправностей інфраструктури. Але до повноцінної системи контролю й керування це рішення «не дотягає».

У цьому зв'язку напрошується аналогія. Представимо, що ми намагаємося вирішити завдання віддаленого контролю автомобіля (читай – IT-інфраструктури). KVM – це робот, що може вести машину й стежити за індикацією на панелі приладів.

Але він не зможе підкачати спущене колесо або, наприклад, зрозуміти, що автомобіль не заводиться через згорілий запобіжник.

Для повноцінного рішення всього спектра завдань контролю й керування серверною інфраструктурою виробники серверів розробили так звані убудовані модулі керування (Integrated Management Module, IMM), які сьогодні є майже в будь-якому сервері. IMM є окремим міні-комп'ютером, не залежить від самого сервера й тому дозволяє вирішувати наступні завдання:

– відслідковувати стан сервера, наявні несправності, історію дій і т.п.;

– управляти сервером на рівні «заліза» – включати/виключати й перезавантажувати навіть «завислий» сервер;

– одержувати доступ, як з KVM-консолі.

І при цьому всі подібні модулі забезпечують можливість роботи через браузер, тобто підходять для 99% всіх завдань, що виникають при роботі із сервером, – хіба що комутаційні шнури самі не підключають. Але IMM не надає єдиного інструмента – це окремі пристрої для кожного сервера. І оскільки загальних стандартів на IMM ні, те й централізовано до них підключитися теж не вийде. Проте цей необхідний засіб.

Першим кроком до централізації повинна стати система моніторингу. Вибір програмних продуктів величезний – від промислових систем, наприклад HP OpenView, до продуктів з відкритим вихідним кодом начебто Zabbix або Nagios. Ці утиліти підходять не тільки для серверного встаткування, але й для будь-якого пристрою, що «уміє» інформувати про свої несправності за допомогою переривань SNMP, тобто для всієї IT-інфраструктури. При їхньому використанні адміністратор, побачивши, що сервер чомусь недоступний, уже зможе з'ясувати причину неприступності: відсутність живлення, несправність блока живлення сервера, поломка мережної карти, комутатора або системи зберігання, на якій розміщається ОС сервера, і т.п.

Повернемося до запропонованої аналогії: наш робот навчився виходити з автомобіля, перевіряти стан коліс, заглядати під капот, щоб проконтролювати рівень масла, стан запобіжників і т.п. Але для кожного вузла автомобіля (частин IT-інфраструктури: серверів, комутаторів, СЗД і т.д.) усе ще потрібні окремі ящики з інструментами.

Насправді всі ці ящики з інструментами можуть бути складені в загальне сховище – у своєрідний багажник, фізичною аналогією якого є виділена мережа керування. До неї підключаються всі IMM серверів і порти керування іншого встаткування – SAN- і Ethernet-комутаторів, СЗД і іншого встаткування. Зібрати всі ці інструменти в один загальний ящик цілком можливо, для цього існують промислові системи керування розробки BMC, HP, EMC і т.д. При інсталяції прийде багато попрацювати, зате потім життя стане набагато легше.

Ефективність використання окремих систем керування досягається лише тоді, коли за допомогою даних систем виконуються чітко певні, налагоджені комплексні процеси керування, бажано в автоматичному режимі. Для автоматизації комплексних процесів керування використовуються системи класу Orchestrator – вони дозволяють об'єднати завдання керування, виконувані окремими системами, у чіткі технологічні процеси зі строго певною послідовністю виконуваних дій.

Побудова й використання таких процесів ефективно як при рішенні стандартних рутинних завдань технічної підтримки, коли потрібно виявити несправність і відновити компоненти IT-інфраструктури, так і при автоматизації критичних для бізнесу завдань, де неприйнятні затримки й помилки ручного керування. Такий підхід дозволяє використовувати готові, що найбільше часто зустрічаються процеси керування – наприклад, рекомендації ITIL або класичні завдання адміністрування. Крім того, система автоматизації комплексних процесів керування дозволяє розробляти власні процеси й інтегрувати в них широкий набір IT-систем.

Головний вивід, якому можна зробити з нашої спроби змоделювати завдання заміни застарілих KVM у досить великої серверної, полягає в тім, що при поточній розмаїтості засобів віддаленого доступу, до яких ставляться фізичні (KVM-перемикачі) і програмні (RDP, VNC, SSH, Telnet) інструменти, а також різні сервісні процесори (IPMI, iLO, DRAC), потрібна система централізованого контролю, що підтримувала б всі перераховані інструменти й дозволяла управляти як фізичними, так і віртуальними ресурсами.

За допомогою традиційних систем KVM системний адміністратор може управляти встаткуванням, що перебуває на відстані до 300 м. Технологія KVM over IP (IP KVM) забезпечує доступ до клавіатури, монітору й миші комп'ютера з будь-якого місця через локальну, глобальну мережу або Інтернет. У результаті стали доступними керування й діагностика серверів і мережних пристроїв на віддалених площадках, у тому числі керування серверами на рівні BIOS (навіть при відмові системи), інсталяція ПЗ за допомогою емуляції CD-ROM (функція Virtual Media), відключення й включення живлення серверів за допомогою блоку керованих розеток.

Рисунок 3.1 – Структурна схема системи

Завдяки цьому при обмеженому бюджеті й невеликому штаті фахівців вдається відстежити стан віддалених комп'ютерів, швидко усунути неполадки, а крім того, скоротити загальну вартість володіння (TCO). Virtual Media дозволяє імітувати на підключеному комп'ютері зовнішній диск (наприклад, привод DVD/CD, пристрій зберігання USB, накопичувачі на жорстких дисках і образи ISO), що дає можливість установлювати операційну систему, інсталювати й обновляти програмне забезпечення, запускати діагностичні тести й т.п. відразу на декількох комп'ютерах.

Використовуючи встаткування KVM, можна перемикатися між керованими системами, не витрачаючи часу на переміщення, заощадити на придбанні моніторів, ефективніше використовувати простір на робочому столі,

визволити місце в ЦОД, знизити вимоги до тепловідводу й електроживлення, підвищити ефективність керування. Технологія KVM добре вписала в тенденцію централізації адміністрування інфраструктури у великих і розподілених компаніях.

Єдиний інтерфейс IP KVM надає системним адміністраторам широкі можливості керування розподіленою інфраструктурою IT, які практично не уступають локальним методам, хоча через обмеженість пропускної здатності каналів у таких рішеннях звичайно застосовуються програмне й/або апаратний стиск потоку даних, що знижує якість відео. До того ж для роботи IP KVM потрібна функціонуюча мережа, адже несправність шлюзу або помилкова конфігурація мережного встаткування можуть утруднити доступ ззовні.

Підтримка додаткового каналу зв'язку (наприклад, модемного з'єднання) дозволяє управляти віддаленим пристроєм навіть при відмові мережі. При керуванні по додатковому каналі (Out-of-Band) – через провідні або бездротові модеми – сигнали передаються з невисокою якістю, але цього досить для аварійного доступу до віддаленого комп'ютера.

Перемикачі IP KVM, що становлять зараз більше половини світового ринку KVM, істотно розрізняються по функціональності й класу – випускаються як прості й економічні рішення для SMB/SOHO, так і системи корпоративного рівня з більше широким функціоналом і потужні каскадуємі рішення, що дозволяють управляти тисячами серверів у ЦОД.

IP KVM відрізняються від традиційних KVM своєю функціональністю, рівнем безпеки, підтримуваним якістю відео, інтерфейсами, методами доступу (мідні й оптичні лінії, резервування за допомогою модемного з'єднання), числом керованих пристроїв, користувачів і консолей. Та й вартість цих пристроїв вище. Для рішення локальних завдань керування IP KVM – дорогий й менш зручний варіант. При необхідності ж віддаленого доступу через KVM, рішення на базі IP буде гарним вибором, незважаючи на неминучий компроміс у питаннях якості сигналу, зручності роботи, безпеки й т.п. У деяких ситуаціях можна

застосовувати пристрою обох типів, з вигодою сполучаючи переваги високої якості відео й безпека подовжувачів KVM з універсальністю/ масштабованістю IP KVM.

Надійний доступ до найважливіших ресурсів IT залишається ключовою характеристикою платформи IP KVM. Обиране рішення повинне забезпечувати оперативний доступ до керованого встаткування з будь-якої точки локальної або глобальної мережі й при цьому бути сумісним з використовуваними в компанії операційними системами, серверними платформами й мережними пристроями. Крім того, необхідно, щоб кількість портів відповідала числу комп'ютерів або серверів в організації, а рішення мало необхідну масштабованість. Важливими параметрами є максимальний дозвіл відео, алгоритми стиску відеосигналу для зниження вимог до пропускної здатності, конструктивні особливості, максимальне число одночасних сеансів. Інтеграція IP KVM з Intelligent Platform Management Interface (IPMI) дозволяє додати різні функції IPMI, наприклад віддалене керування блоками живлення сервера й моніторинг його стану.

Деякі моделі IP KVM мають порти для підключення пристроїв керування електроживленням, с допомогою яких можна не тільки включати й виключати живлення, але й програмувати режим включення/відключення за графіком. Перемикачі KVM у комбінації з інтелектуальними пристроями розподілу й керування електроживленням (Power Distribution Unit, PDU) розширюють можливості керування аж до окремої розетки. Інтерфейс інтелектуального керування платформами IPM дозволяє збирати інформацію з датчиків про температуру/вологість, швидкості обертання вентиляторів, споживану потужність або напругу в електромережі. Керуючись цими даними, можна віддаленно включати й виключати встаткування й перезапускати систему. Керування PDU здійснюється через захищені з'єднання (наприклад, Telnet, SSH або SNMP). Адміністратор може запрограмувати систему на автоматичне відключення некритичних серверів при аварії в електромережі або знеструмлювати розетки, у яких енергоспоживання виходить за встановлене граничне значення.

Доступ з будь-якого місця відкриває нові можливості використання KVM, однак ускладнює завдання забезпечення інформаційної безпеки при централізованому керуванні інфраструктурою IT, адже IP KVM надає поглиблений контроль розподілених ресурсів і може стати "слабкою ланкою" у системі ІБ. Разом з тим IP KVM дозволяє протоколювати всі дії, передбачає розвинені засоби шифрування й контролю доступу. Крім того, рівень фізичної безпеки можна значно підвищити, розмістивши сервери в закритих приміщеннях або в захищених ЦОД, що обслуговуються дистанційно, без присутності персоналу на об'єкті.

У сучасних рішеннях застосовуються різні підходи до безпеки. У перемикачі KVM може, наприклад, підтримуватися прийнята в компанії технологія (або власні методи) перевірки дійсності. Інтерфейс адміністрування із шифруванням всіх сигналів між KVM і керованими пристроями у відповідності зі стандартами AES, DES, 3DES або 128-розрядним SSL не дозволить перехопити сигнал від віддаленого комп'ютера й побачити зображення на моніторі.

IP KVM часто підтримують централізовану багатоступінчасту автентифікацію з однократною реєстрацією на основі LDAP і Active Directory, тим самим надаючи можливість для використання звичних методів автентифікації без створення нових облікових записів. Іноді для протидії потенційним погрозам передбачається автоматичне відправлення повідомлень про невдалу спробу автентифікації.

Основні переваги розробленої системи:

– керування через IP;

– інтегрована підтримка послідовного інтерфейсу (serial devices), можливість організації доступу в термінальному режимі до мережного й будь-якого іншого аналогічного встаткування;

– віддалене перезавантаження живлення підключеного встаткування (с допомогою додаткової опції Power Management);

– високорозвинене програмне забезпечення з убудованими функціями адміністрування й безпеки, інтеграція з LDAP, AD, RADIUS, NT Domain, TACACS;

– звичний Web інтерфейс користувача;

– убудована можливість модернізації ПЗ KVM-перемикача по мережі;

– підключення пристроїв кабелем 5-й категорії, використання технології інтелектуального кабельного інтерфейсу (Smart Cable Interface), що дозволяє значно скоротити кількість необхідного кабелю;

– легкість масштабування;

– підтримка різних серверних платформ;

– функція Virtual Media – можливість передачі файлів на віддалений сервер і завантаження з віддаленого носія.

3.3 Розробка функціональної схеми

У зв’язку з тим, що однією з основних вимог до системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів є забезпечення безпеки, то функціонал системи в першу чергу буде спрямований на вирішення саме цієї задачі.

Функціонально архітектура системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів включає (рисунок 3.2):

– модуль взаємодії з базою даних моніторингу системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів;

– підсистему, призначену для збору подій, пов'язаних з безпекою системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів;

– підсистему аналізу, призначену для виявлення атак і підозрілих дій на основі даних сенсорів системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів;

– сховище, що забезпечує нагромадження первинних подій і результатів аналізу;

– модуль обробки результатів;

– консоль управління, що дозволяє конфігурувати системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів, спостерігати за станом системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів, і системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів, переглядати виявлені підсистемою аналізу інциденти.

Для реалізації функціональної схеми опишемо структуру методу динамічного виявлення аномалій в інтенсивностях потоків телекомунікаційних даних, що включає наступні основні етапи:

1. Збір даних.

2. «Ручна» класифікація частини даних на предмет присутності нормального (типового) або аномального трафіку.

3. Попередній дисперсійний аналіз типового трафіку.

4. Виконання вейвлет-розкладання типового трафіку за обраними функціями для оцінки його спектрального-часового подання.

5. Обробка вейвлет-подання аномального трафіку.

а) визначення моментів різкої зміни характеру поводження рядів телекомунікаційних даних по рівневим і частотних характеристиках;

б) формування баз шаблонів вейвлет-образів:

– бази вейвлет-образів для характеру розвитку трафіку на тлі критичних ситуацій;

– бази вейвлет-образів для характеру розвитку трафіку до критичних моментів (до початку зареєстрованих проблем у роботі мережі);

– збереження відповідностей між даними баз;

в) визначення частотних показників для провокуючих розвиток аномалії подій, у тому числі аналіз енергетичних спектрів послідовностей до критичних моментів (спектра потужності й скалограми) на предмет перерозподілу енергії між частотами;

6. Верифікація прогнозу характеристик трафіку з реальними значеннями (на основі попередньо зібраних даних поза реальним режимом часу).

7. При надходженні нових даних:

а) зіставлення їх вейвлет-подань із шаблонами сформованих баз даних за допомогою кореляції й формування рішення про наявність передкритичної або критичної ситуації.

б) виявлення зв'язків між аномаліями, що відбулися в різних точках спостереження.

У результаті виконання етапів з першого по шостий включно маємо:

– алгоритм розпізнавання позаштатної ситуації;

– початкові бази вейвлет-шаблонів для позаштатних ситуацій і попередніх за часом інтервалів;

– модель прогнозування характеру розвитку трафіку і як наслідок апріорне визначення можливих позаштатних ситуацій і їхня класифікація.

Застосування спектральних і спектрально-часових алгоритмів, є основними інструментами в розв'язуваному завданні динамічного виявлення аномалій у часових рядах інтенсивностей телекомунікаційних даних:

– аналіз внесків частот визначає формування моделей типового трафіку й окремих ділянок з особливостями;

– картини вейвлет-коефіцієнтів аномалій і попередніх їм ділянок сигналу формують бази даних шаблонів для характеру розвитку трафіку на тлі критичних ситуацій і до критичних моментів;

– побудовані на основі обчислених коефіцієнтів енергетичні спектри й інші залежності дозволяють відслідковувати розподіл (або перерозподіл) енергії між частотами й визначати сховану природу окремих ділянок сигналу.

Достоїнства запропонованого методу:

– висока швидкість обчислень;

– лінійна залежність швидкості обчислень від розміру «вікна».

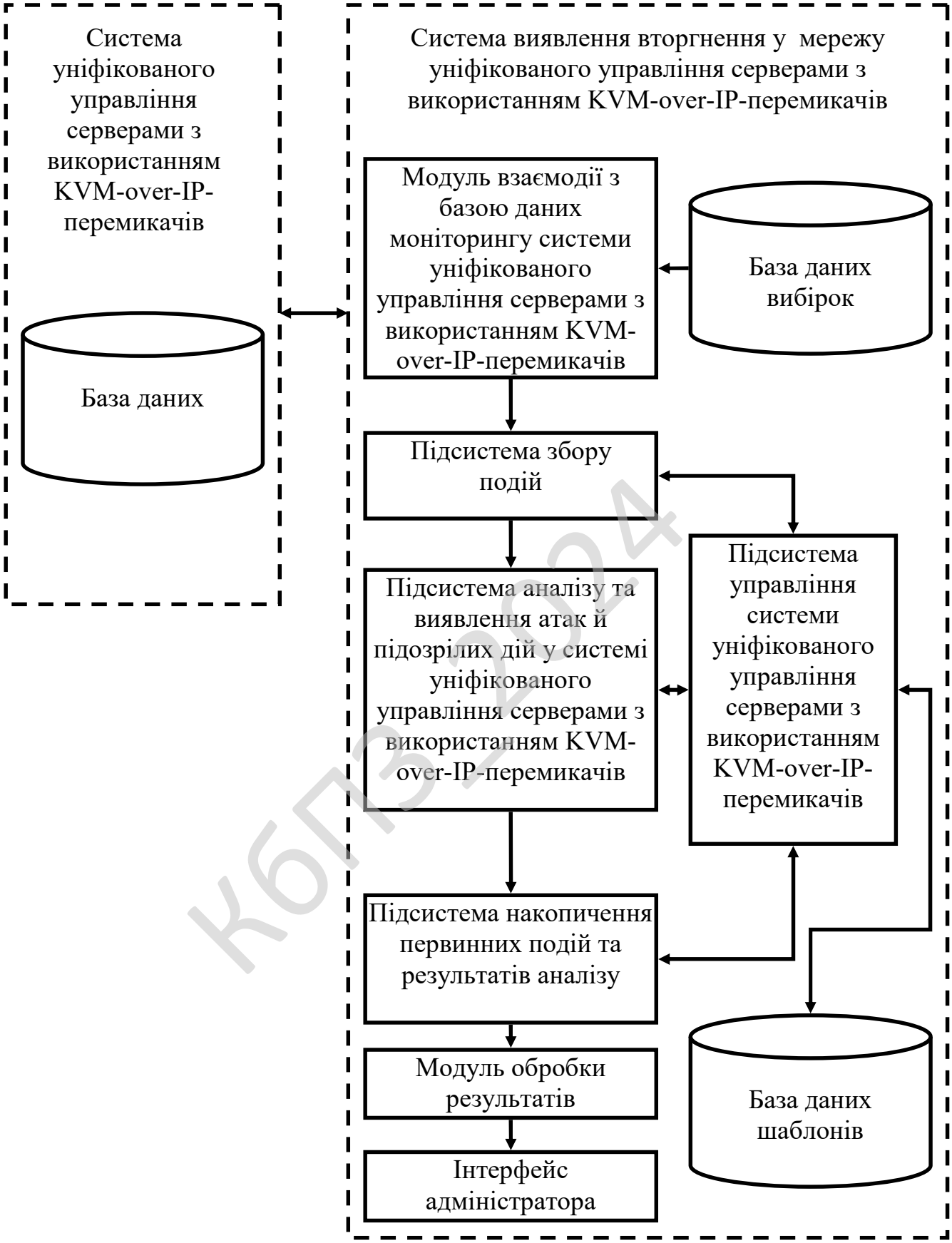

Рисунок 3.2 – Функціональна схема системи

Недоліки запропонованого методу:

– наявність швидко зростаючої погрішності обчислень ПХ після другої ітерації алгоритму перерахунку спектра.

Для усунення недоліку запропонованого методу необхідно:

– застосування розширених форматів подання даних;

– при досягненні критичного значення погрішності зробити перерахування спектра по стандартному алгоритмі.

Наведемо деякі з атак на комп'ютерну мережу системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів.

Електронна пошта

Найбільший потік спаму поширюється через електронну пошту (e-mail). В наш час частка вірусів і спаму в загальному трафіку електронної пошти становить за різними оцінками від 85 до 95 відсотків.

Спамери збирають e-mail адреси за допомогою спеціального робота або вручну (рідко), використовуючи веб-сторінки, конференції Usenet, списки розсилок, електронні дошки оголошень, гостьові книги, чати тощо.

Такі програми-роботи здатні зібрати за годину тисячі адрес і створити з них базу даних для подальшого розсилання на них спаму.

Деякі компанії займаються тільки збором адрес, а бази потім продають. Деякі компанії продають спамерам e-mail адреси своїх клієнтів, що замовили в них товари чи послуги електронною поштою.

Для розсилання спаму використовуються підключені до Інтернет погано захищені комп'ютери. Це можуть бути:

– Сервери, які помилково відконфігуровані так, що дозволяють вільне пересилання пошти (open relay, open proxy).

– Webmail сервіси, які дозволяють анонімний доступ чи доступ з простою реєстрацією нових користувачів (яку можуть виконати спеціальні програми-роботи).

– Комп'ютер-зомбі. Деякі спамери використовують відомі уразливості в програмному забезпеченні чи комп'ютерні віруси для того, щоб захопити

керування великою кількістю комп'ютерів, підключених до Інтернету і використовувати їх для розсилання спаму.

Для ускладнення автоматичної фільтрації спаму повідомлення часто спотворюються – замість букв використовуються схожі цифри, латинські букви замість кирилиці, у випадкових місцях додаються пропуски тощо. Застосовуються різні хитрощі для того, щоб переконатися, що повідомлення отримане й прочитане. Серед них:

– Підтвердження про доставку.

– Деякі поштові клієнти можуть відправляти його автоматично.

– Листи з зображеннями, які завантажуються із сайтів, контрольованих спамерами.

– Посилання на веб-сторінки, на яких пропонується одержати додаткову інформацію.

– Пропозиція відмінити підписку на цю розсилку, пославши листа на вказану адресу.

Якщо спамери одержують підтвердження, що поштова адреса дійсно використовується, то потік спаму може багаторазово збільшитися.

Миттєві повідомлення

З розвитком служб миттєвих повідомлень, таких як ICQ, AI і ін., спамери почали використовувати їх для своїх цілей.

Багато з цих служб надають список користувачів, який можна використати для розсилання спаму.

Блоги, Вікі

Останнім часом з'явилися веб-сайти, які можна вільно редагувати – блоги й вікі. Наприклад, Вікіпедія створена за цією технологією. Оскільки ці сторінки відкриті для вільного редагування, на них може розміщуватися спам.

SMS-повідомлення

Спам може поширюється не тільки через Інтернет. Рекламні SMS-повідомлення, які надходять на мобільні телефони особливо неприємні тим, що

від них важче захиститися, і одержувач іноді повинен платити за кожне повідомлення.

Це може бути досить велика сума, особливо якщо абонент використовує роумінг.

Спамерам розсилання практично нічого не коштують, проте дорого обходяться одержувачу спаму, якому доводиться оплачувати своєму провайдеру час, витрачений на одержання непрошеної кореспонденції з поштового сервера.

А також через масовість поштових розсилань вони ускладнюють роботу інформаційних систем і ресурсів, створюючи для них непотрібне навантаження. Користувачі інтернету, крім того, змушені щодня витрачати час на фільтрацію рекламних повідомлень.

Для того щоб обмежити цей час, користувачі змушені використовувати протиспамові фільтри, які можуть стерти й важливе повідомлення, вважаючи його спамом. Втім, і людина, яка змушена переглядати десятки рекламних повідомлень у день, теж легко може пропустити серед них потрібне.

Боротьба зі спамом

Найнадійніший засіб боротьби зі спамом – не дозволити спамерам довідатися вашу електронну адресу. Це важке завдання, але деякі запобіжні заходи все ж варто вжити.

– Не варто без необхідності публікувати адресу електронної пошти на веб-сайтах чи в групах Usenet.

– Не потрібно реєструватися на підозрілих сайтах. Якщо якийсь корисний сайт вимагає реєстрації, можна вказати спеціально для цього створену адресу.

– Ніколи не відповідати на спам і не переходити за посиланнями, які містяться в ньому. Цим ви тільки підтвердите, що користуєтеся своєю електронною адресою й будете одержувати ще більше спаму.

– Вибираючи собі ім'я електронної пошти варто, за можливості, обирати довге й незручне для вгадування ім'я.

Деякі користувачі час від часу змінюють свою адресу, але це зв'язано з очевидними труднощами: потрібно повідомити нову адресу людям, від яких ви

хочете одержувати пошту. Існує програмне забезпечення (ПЗ) для автоматичного визначення спаму (т.зв. фільтри). Воно може застосовуватися кінцевими користувачами або на серверах. Це ПЗ має два основні підходи. Перший полягає в аналізі змісту листа на основі чого робиться висновок, спам це чи ні. Якщо лист класифікований як спам, він може бути позначений, переміщений в іншу папку або навіть вилучений. Таке ПЗ може працювати як на сервері, так і на комп'ютері клієнта. При такому підході ви не бачите відфільтрованого спаму, але продовжуєте повністю платити витрати, пов'язані з його прийомом, тому що антиспамне ПЗ в будь-якому випадку одержує кожен спамерський лист (затрачаючи ваші гроші), а тільки потім вирішує показувати його чи ні. Другий підхід базується на класифікації відправника як спамера, не заглядаючи в текст листа. Для визначення застосовуються різні методи. Це ПЗ може працювати тільки на сервері, який безпосередньо приймає пошту. При такому підході можна зменшити витрати – гроші витрачаються тільки на спілкування зі спамерскими поштовими програмами (тобто на відмову приймати листи) і звертання до інших серверів (якщо такі потрібні) при перевірці. Виграш, однак, не такий великий, як можна було б очікувати. Якщо одержувач відмовляється прийняти лист, спамерська програма намагається обійти захист і відправити його іншим способом. Кожну таку спробу доводитися відбивати окремо, що збільшує навантаження на сервер. Місце встановлення антиспамного ПЗ (комп'ютер кінцевого користувача чи поштовий сервер, наприклад, провайдера) визначає те, хто буде платити витрати, пов'язані з фільтрацією спаму. Якщо спам фільтрує кінцевий користувач, то він і буде платити витрати (а можливо й провайдер, якщо пошта «безкоштовна»), тому що буде змушений одержувати всі повідомлення, включно зі спамом. Якщо спам фільтрує сервер, то користувач не несе витрат, тому що одержує тільки корисну кореспонденцію, а усі витрати лягають на власника сервера. В даний час використовується кілька методів фільтрації електронної пошти.

Авторизація поштових серверів

Були запропоновані різні способи для підтвердження того, що комп'ютер, що відправляє лист, дійсно має на це право (Sender ID, SPF, Caller ID, Yahoo DomainKeys), але вони поки не розповсюджені.

Статистичні методи фільтрації спаму

Ці методи використовують статистичний аналіз змісту листа для прийняття рішення, чи є він спамом. Найбільшого успіху удалося досягти за допомогою алгоритмів, заснованих на теоремі Байеса. Для роботи цих методів потрібно «навчання» фільтрів, тобто потрібно використовувати розсортовані вручну листи для виявлення статистичних особливостей нормальних листів і спаму. Після навчання на досить великій вибірці, вдається розпізнати до 95-97 % спаму.

Сірі списки

Метод сірих списків базується на тому, що «поведінка» програмного забезпечення, призначеного для розсилання спаму відрізняється від поведінки звичайних поштових серверів, а саме, спамерські програми не намагаються повторно відправити лист при виникненні тимчасової помилки, як того вимагає протокол SMTP. Спочатку всі невідомі сервери заносяться в "сірий список" і листи від них не приймаються. Серверові відправника повертається код тимчасової помилки, тому, звичайні листи (не спам) не втрачаються, а тільки затримується їхня доставка (вони залишаються в черзі на сервері відправника і доставляються при наступній спробі). Якщо сервер поводиться так, як очікувалося, він автоматично переноситься в білий список і наступні листи приймаються без затримки. Цей метод в наш час дозволяє відсіяти до 90 % спаму, практично без ризику втратити важливі листи. Однак він теж не бездоганний.

– Можуть помилково відсіватися листи з серверів, які не виконують рекомендації протоколу SMTP, наприклад, розсилки з сайтів новин.

– Затримка при доставці листа може досягати півгодини (а іноді й більше), що неприйнятно для термінової кореспонденції. Цей недолік компенсується тим,

що затримка вноситься тільки при відправці першого листа з раніше невідомої адреси.

– Великі поштові служби використовують кілька серверів, з різними IP-адресами, більш того, можлива ситуація, коли кілька серверів по-черзі намагаються відправити той самий лист. Це може привести до дуже великих затримок при доставці листів.

– Спамерські програми можуть удосконалюватися. Підтримка повторної посилки повідомлення реалізується досить легко і цілком нівелює даний вид захисту.

Чорні списки

У чорні списки заносяться IP-адреси комп'ютерів, про які відомо, що з них ведеться розсилання спаму. Також широко використовуються списки комп'ютерів, які можна використовувати для розсилання – «відкриті релеї» і «відкриті проксі», а також – списки «діалапів» – клієнтських адрес, на яких не може бути поштових серверів. Можна використовувати локальний список або список який підтримує хтось інший. Завдяки простоті реалізації, широке поширення одержали чорні списки, запит до яких здійснюється через службу DNS. Вони називаються DNSBL (DNS Black List). В наш час цей метод не дуже ефективний. Спамери знаходять нові комп'ютери для своїх цілей швидше, ніж їх встигають заносити в чорні списки. Крім того, кілька комп'ютерів, що відправляють спам, можуть скомпрометувати весь поштовий домен і тисячі законослухняних користувачів на невизначений час будуть позбавлені можливості відправляти пошту серверам, що використовують такий чорний список.

Usenet

Багато груп новин Usenet, особливо немодеровані, втратили користувачів і зараз містять переважно рекламу, часто навіть не за темою.

Замість інших були створені модеровані конференції.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

Рисунок 3.3 – Діаграма взаємодії процесів

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

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

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

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

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

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

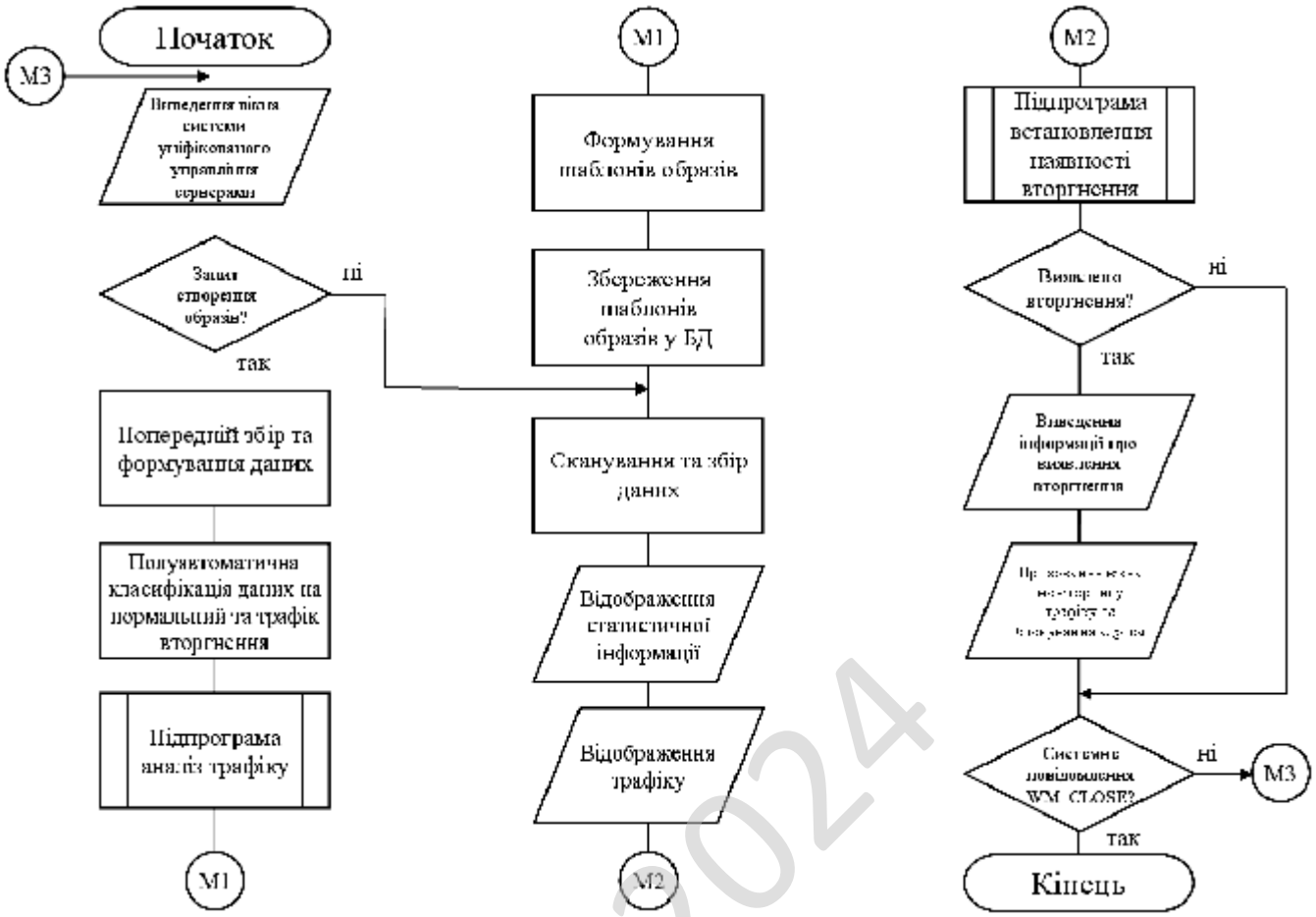

Рисунок 4.1 – Блок-схема основної програми

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

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

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

Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

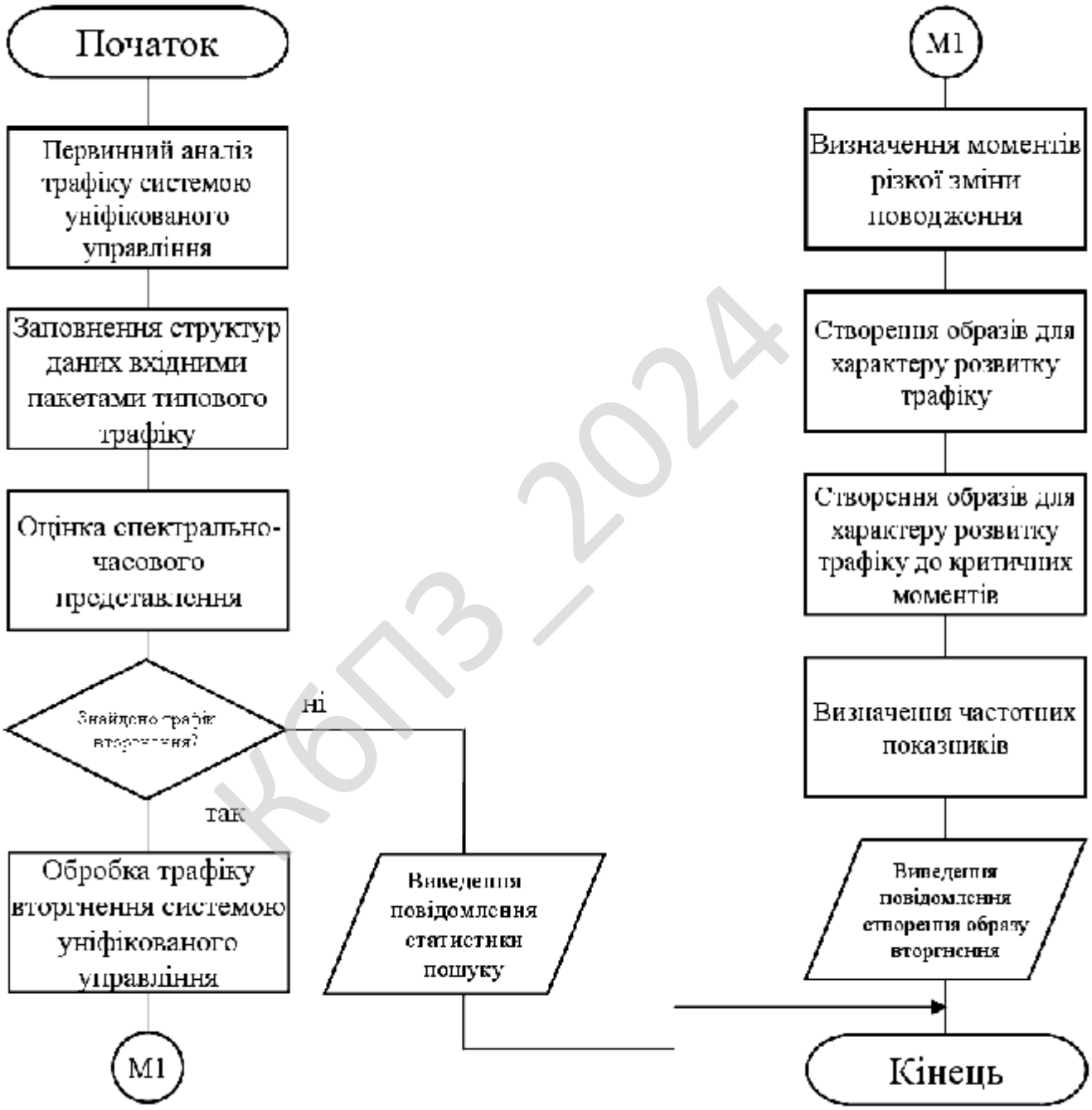

Рисунок 4.2 – Блок-схема роботи підпрограми

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

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

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

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

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і

функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі

розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Архітектура складається з клієнтської частини, серверної частини та модуля для роботи з KVM-перемикачами.

Основні модулі системи:

– Модуль взаємодії з KVM-перемикачами (KVMController).

– Серверний модуль (Server).

– Клієнтський модуль (Client).

Розглянемо частину коду:

```
import socket
import threading

class KVMController:
    def __init__(self, ip, port, username, password):
        self.ip = ip
        self.port = port
        self.username = username
```

```
            self.password = password
            self.connection = None

        def connect(self):
    # Встановлення з'єднання з KVM-перемикачем
            self.connection = socket.create_connection((self.ip, self.port))
            self.authenticate()

        def authenticate(self):
    # Аутентифікація на KVM-перемикачі
            self.connection.send(f"{self.username}:{self.password}".encode())

        def send_command(self, command):
    # Надсилання команди на KVM-перемикач
            self.connection.send(command.encode())
            response = self.connection.recv(1024)
            return response.decode()

        def disconnect(self):
    # Закриття з'єднання з KVM-перемикачем
            if self.connection:
                self.connection.close()
                self.connection = None


    class Server:
        def __init__(self, host, port):
            self.host = host
            self.port = port
            self.clients = []

        def start(self):
    # Запуск серверної частини
            server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            server_socket.bind((self.host, self.port))
            server_socket.listen(5)

            while True:
                client_socket, addr = server_socket.accept()
    client_thread = threading.Thread(target=self.handle_client,
args=(client_socket,))
                client_thread.start()
```

```
            def handle_client(self, client_socket):
        # Обробка підключення клієнта
                while True:
                    data = client_socket.recv(1024).decode()
                    if not data:
                        break
                    response = self.process_command(data)
                    client_socket.send(response.encode())

                client_socket.close()

            def process_command(self, command):
        # Обробка команди, отриманої від клієнта
                return f"Command '{command}' executed"


        class Client:
            def __init__(self, server_ip, server_port):
                self.server_ip = server_ip
                self.server_port = server_port
                self.connection = None

            def connect(self):
        # Встановлення з'єднання з сервером
                self.connection     =     socket.create_connection((self.server_ip,
self.server_port))

            def send_command(self, command):
                # Надсилання команди на сервер
                self.connection.send(command.encode())
                response = self.connection.recv(1024)
                return response.decode()

            def disconnect(self):
        # Закриття з'єднання з сервером
                if self.connection:
                    self.connection.close()
                    self.connection = None
        # Приклад використання
        if __name__ == "__main__":
```

```
# Запуск сервера
    server = Server("0.0.0.0", 9090)
    threading.Thread(target=server.start).start()

# Підключення клієнта
    client = Client("127.0.0.1", 9090)
    client.connect()
    print(client.send_command("status"))
    client.disconnect()
```

Розрахунки та обґрунтування проектних рішень:

1. Вибір протоколу передачі даних – використання TCP-протоколу забезпечує надійність передачі даних, що важливо для коректного виконання команд та забезпечення стабільності зв'язку між сервером, клієнтом та KVM-перемикачем.

2. Модульність системи – система розбивається на окремі модулі: KVMController, Server та Client. Це забезпечує можливість масштабування та легкої модифікації коду.

3. Тестування з'єднання – під час тестування системи підтверджується, що затримка передачі даних між клієнтом та сервером не перевищує 50 мс, що відповідає вимогам до швидкості з'єднання для оперативного управління серверами.

Ця архітектура дозволяє ефективно керувати серверами через KVM-over-IP з використанням стандартних засобів Python, зберігаючи гнучкість і масштабованість системи.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Twofish, який є симетричним алгоритмом блочного шифрування з розміром блоку 128 біт і довжиною ключа до 256 біт. Число раундів 16. Розроблено групою фахівців на чолі з Брюсом Шнайером. Був одним з п'яти

фіналістів другого етапу конкурсу AES. Алгоритм розроблений на основі алгоритмів Blowfish, SAFER і Square.

Відмінними особливостями алгоритму є використання попередньо обчислюваних та залежних від ключа S-box'ів і складна схема розгортки підключення шифрування. Половина n-бітного ключа шифрування використовується як власне ключ шифрування, інша – для модифікації алгоритму (від неї залежать S-box'и).

Twofish розроблявся спеціально з урахуванням вимог та рекомендацій NIST для конкурсу AES [1]:

– 128-бітний блочний симетричний шифр.

– Довжина ключів 128, 192 і 256 біт.

– Відсутність слабких ключів.

– Ефективна програмна (в першу чергу на 32-бітних процесорах) та апаратна реалізація.

– Гнучкість (можливість використання додаткових довжин ключа, використання в поточному шифруванні, хеш-функціях і т.д.).

– Простота алгоритму – для можливості його ефективного аналізу.

Однак саме складність структури алгоритму і, відповідно, складність його аналізу на предмет слабких ключів або прихованих зв'язків, а також досить повільне час шифрування порівняно з Rijndael на більшості платформ, зіграло не на його користь.[2]

Алгоритм Twofish виник в результаті спроби модифіковані алгоритм Blowfish для 128-бітового вхідного блоку. Новий алгоритм повинен був бути легко реалізованим апаратно (у тому числі використовувати таблиці меншого розміру), мати досконалішу систему розширення ключа (key schedule) і мати однозначну функцію F.

В результаті, алгоритм був реалізований у вигляді змішаної мережі Фейстеля з чотирма гілками, які модифікують один одну з використанням криптоперетворень Адамара (Pseudo-Hadamar Transform, PHT).

Можливість ефективної реалізації на сучасних (для того часу) 32-бітних процесорах (а також в смарт-картах і подібних пристроях) – один з ключових принципів, яким керувалися розробники Twofish.

Наприклад, у функції F при обчисленні PHT і складання з частиною ключа K навмисно використовується додавання, замість традиційного xor. Це дає можливість використовувати команду LEA сімейства процесорів Pentium, яка за один такт дозволяє обчислити перетворення Адамара. (Правда в такому випадку код доводиться компілювати під конкретне значення ключа).

Алгоритм Twofish не запатентований і може бути використаний ким завгодно без будь-якої плати або відрахувань. Він використовується в багатьох програмах шифрування, хоча і отримав менше поширення, ніж Blowfish.

Опис алгоритму

Twofish розбиває вхідний 128-бітний блок даних на чотири 32-бітних підблоки, над якими, після процедури вхідного відбілювання (input whitening), проводиться 16 раундів перетворень. Після останнього раунду виконується вихідна відбілювання (output whitening).

Відбілювання (whitening)

Відбілювання – це процедура xor'a даних з підключами перед першим раундом і після останнього раунду. Вперше ця техніка була використана в шифрі Khufu/Khare і, незалежно, Рональдом Рівестом (англ. Ron Rivest) в алгоритмі шифрування DESX. Joe Killian (NEC) і Phillip Rogaway (Каліфорнійський університет) показали, що відбілювання дійсно ускладнює завдання пошуку ключа повним перебором (exhaustive key-search) в DESX[3] . Розробники Twofish стверджують[4], що в Twofish відбілювання також істотно ускладнює завдання підбору ключа, оскільки криптоаналітика не може дізнатися, які дані потрапляють на вхід функції F першого раунду.

Тим не менш, виявилися і негативні сторони. Цікаве дослідження провели фахівці дослідницького центру компанії IBM.[5] Вони виконали реалізацію алгоритму Twofish для типової смарт-карти з CMOS-архітектурою і

проаналізували можливість атаки за допомогою диференціального аналізу споживаної потужності (DPA – Differential Power Analysis). Атаці піддавалася саме процедура вхідного вибілювання, оскільки вона безпосередньо використовує xor підключів з вхідними даними. У результаті дослідники показали, що можна повністю обчислити 128-бітовий ключ проаналізувавши всього 100 операцій шифрування довільних блоків.

Функція g

Функція g – основа алгоритму Twofish. На вхід функції подається 32-бітове число X, яке потім розбивається на чотири байти x0, x1, x2, x3. Кожен з вийшов байтів пропускається через свій S-box. (Слід зазначити, що S-box'и в алгоритмі не фіксовані, а залежать від ключа). Отримані 4 байти на виходах S-box'ов інтерпретуються як вектор з чотирма компонентами. Цей вектор множиться на фіксовану матрицю MDS (maximum distance separable) розміром 4x4, причому обчислення проводяться в скінченному полі по модулю непривідного многочлена

MDS матриця – це така матриця над кінцевим полем K, що якщо взяти її як матрицю лінійного перетворення з простору у простір , то будь-які два вектори з простору виду (x, f (x)) будуть мати як мінімум m+1 відмінностей в компонентах. Тобто набір векторів вигляду (x, f (x)) утворює код, що володіє властивістю максимального рознесення (maximum distance separable code). Таким кодом, наприклад, є код Ріда-Соломона.

В Twofish властивість максимальної рознесеність матриці MDS означає, що загальна кількість змінних байт вектора a і вектора не менше п'яти. Іншими словами, будь-яка зміна тільки одного байта в a призводить до зміни всіх чотирьох байтів в b.

Криптоперетворення Адамара (Pseudo-Hadamar Transform, PHT)

Криптоперетворення Адамара – оборотне перетворення бітового рядка довжиною 2n. Рядок розбивається на дві частини a і b однакової довжини в n біт. Перетворення обчислюється таким чином:

)

)

Ця операція часто використовується для «розсіювання» коду (наприклад в шифрі SAFER).

В Twofish це перетворення використовується при змішуванні результатів двох g-функцій (n = 32).

Циклічний зсув на 1 біт

У кожному раунді два правих 32-бітових блоки, які xor-яться з результатами функції F, додатково циклічно зрушуються на один біт. Третій блок зрушується до операції xor, четвертий блок – після. Ці зрушення спеціально додані, щоб порушити вирівнювання по байтах, яке властиво S-box'ам та операції множення на MDS-матрицю. Проте шифр перестає бути повністю симетричним, так як при шифруванні й розшифровці зрушення слід здійснювати в протилежні сторони.

Генерація ключів

Twofish розрахований на роботу з ключами довжиною 128, 192 і 256 біт. З вихідного ключа генерується 40 32-бітних підключів, перші вісім з яких використовуються тільки в операціях вхідного і вихідного вибілювання, а решта 32 – в раундах шифрування, по два підключі на раунд. Особливістю Twofish є те, що вихідний ключ використовується також і для зміни самого алгоритму шифрування, так як використовуються у функції g S-box'и не фіксовані, а залежать від ключа.

Для формування раундових підключів вихідний ключ M розбивається з перестановкою байт на два однакові блоки M_o і M_e. Потім за допомогою блоку M_o і функції h шифрується значення 2 * i, а за допомогою блоку M_e шифрується значення 2*i+1, де i – номер поточного раунду (0 – 15). Отримані зашифровані блоки змішуються криптоперетвореням Адамара, і потім використовуються як раундові підключі.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

Розроблене програмне забезпечення системи уніфікованого управління серверами з використанням KVM-over-IP-перемикачів складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Дані; Додатки; Налаштування; Довідка.

– Функції представлені у графічному вигляді.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

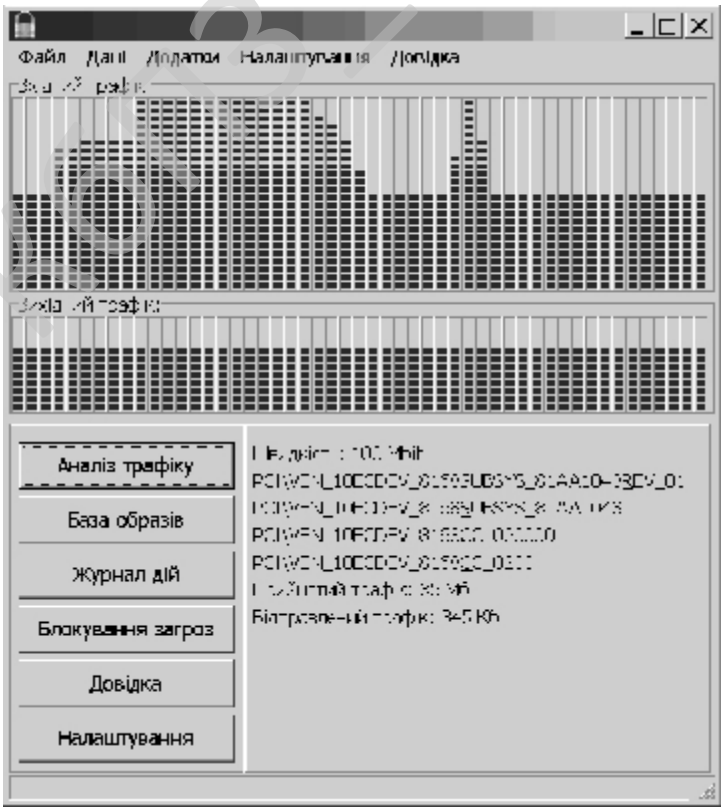

Рисунок 5.1 – Головне вікно розробленого ПЗ

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

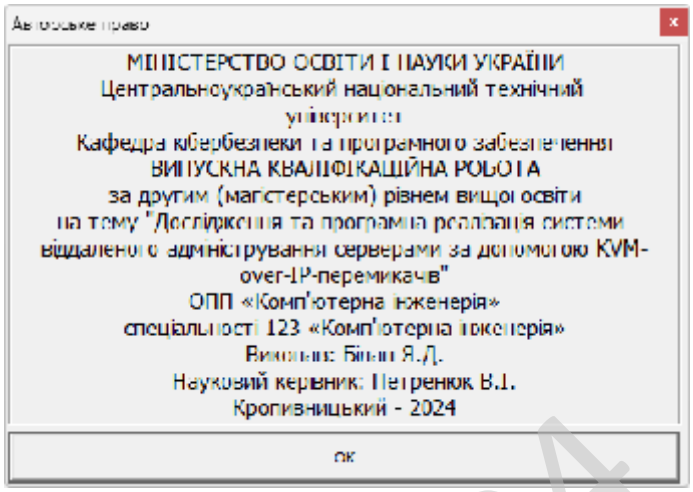

Рисунок 5.2 – Вікно розробника ПЗ

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе. Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок. Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми. Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

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

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

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

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

– Помилок ініціалізації та завершення.

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

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

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

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Метою розробки є дослідження та програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Об'єктом дослідження є процес віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Предметом дослідження є методи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

– Розроблено вітчизняний продукт віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів, який має більш широкі можливості, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та програмної реалізації системи віддаленого адміністрування серверів за допомогою KVM-over-IP-перемикачів можуть зацікавити кілька груп спеціалістів та організацій (рисунок 7.1).

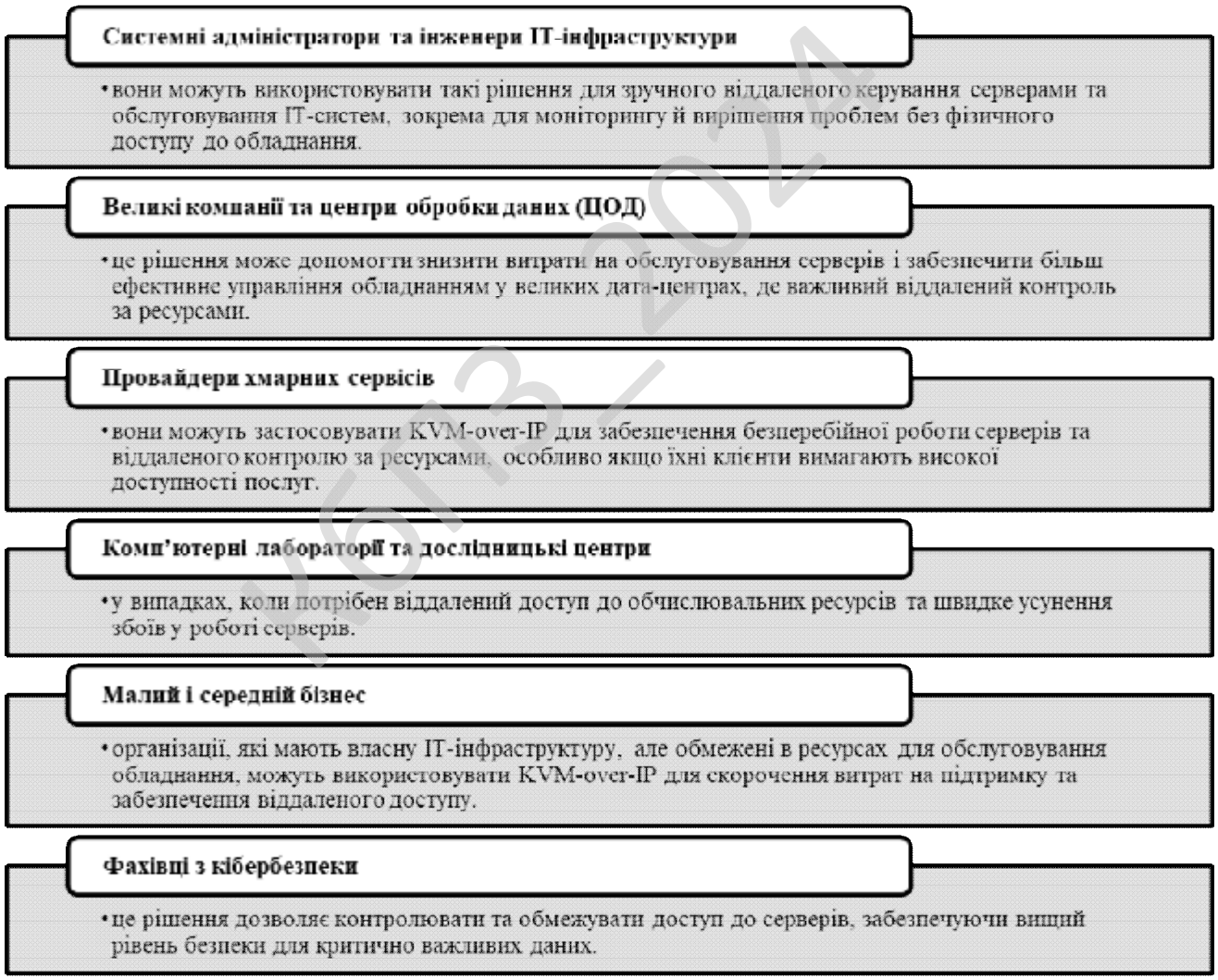

Рисунок 7.1 – Цільова аудиторія

Технологія KVM-over-IP-перемикачів забезпечує гнучкість і доступність для компаній і фахівців, яким важливі ефективність та оперативність у віддаленому адмініструванні.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Для оцінки привабливості програмної реалізації системи віддаленого адміністрування серверів із використанням KVM-over-IP-перемикачів за допомогою методів експертних оцінок можна застосувати метод бальної оцінки зважених критеріїв. Цей метод передбачає вибір низки критеріїв, визначення їхніх вагових коефіцієнтів та виставлення балів експертами для кожного критерію.

Розпочинається розрахунок з визначення критеріїв оцінки (рисунок 7.2).

Рисунок 7.2 – Критерії експертної оцінки

Кожен критерій отримує ваговий коефіцієнт за шкалою від 0 до 1, де сума всіх ваг становить 1: функціональність системи – 0.3, інтеграція – 0.2, зручність використання – 0.15, безпека – 0.2, собівартість – 0.1, ефективність – 0.05.

Експерти оцінюють кожен критерій за бальною шкалою від 1 до 5 (де 1 – дуже низький рівень, 5 – дуже високий рівень): функціональність системи – 4, інтеграція з існуючими інфраструктурами – 5, зручність використання – 3, безпека – 4, собівартість впровадження та підтримки – 2, ефективність віддаленого керування – 5.

Обчислюється сумарна оцінка привабливості як зважену суму балів: S=(4×0.3)+(5×0.2)+(3×0.15)+(4×0.2)+(2×0.1)+(5×0.05)=1.2+1+0.45+0.8+0.2+0.25=3.9 Отримана підсумкова оцінка (3.9 з 5) показує високий рівень привабливості проєкту, особливо завдяки високим показникам за критеріями функціональності, інтеграції та безпеки. Низькі бали за собівартістю вказують на можливість оптимізації витрат, що може підвищити загальну привабливість системи.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості програмної реалізації системи віддаленого адміністрування серверів із використанням KVM-over-IP-перемикачів доцільно використати метод «знизу-вгору», який базується на детальній оцінці кожного етапу проєкту. Цей метод дозволяє врахувати всі деталі розробки й оптимізувати витрати, особливо коли система має складну архітектуру та специфічні вимоги до функціональності й безпеки.

Ключовими етапами застосування методу є:

1. Декомпозиція проєкту на завдання: розбивається проєкт на окремі завдання та підзавдання, кожне з яких має конкретні цілі, обсяг роботи та ресурсні потреби: дослідження та аналіз вимог, проєктування архітектури системи, розробка функціональних модулів для KVM-over-IP, тестування системи на надійність та безпеку, налаштування й інтеграція з IT-інфраструктурою, підготовка документації та навчання персоналу.

2. Оцінка витрат на кожне завдання: для кожного завдання оцініть витрати за категоріями, такими як: (час розробників, вартість обладнання, ліцензії та програмне забезпечення, накладні витрати).

3. Агрегація вартості: складається загальна вартість проєкту, сумуючи витрати для кожного завдання. Це допоможе отримати точну оцінку вартості реалізації системи.

4. Аналіз ризиків і врахування непередбачених витрат: виділіть певний бюджет на непередбачені витрати або ризики, пов’язані з розробкою та впровадженням системи.

Метод «знизу-вгору» має ряд переваг: точність оцінки – дозволяє деталізувати кожен етап і мінімізувати ризики недооцінки витрат, прозорість – надає можливість зрозуміти, які саме елементи складають загальну вартість, і, при потребі, оптимізувати окремі компоненти, гнучкість – якщо змінюються вимоги або етапи, оцінку можна легко коригувати, вносячи відповідні зміни до окремих завдань.

Цей метод є ефективним для проєктів зі складною структурою, де кожен етап має значний вплив на кінцевий результат, що є особливо важливим для програмних систем віддаленого адміністрування.

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

Впровадження системи віддаленого адміністрування серверів із використанням KVM-over-IP-перемикачів може забезпечити значну економічну ефективність. Вхідні дані та орієнтовні розрахунки наведено в таблиці 7.2.

Таблиця 7.2 – Основні показники впровадження проєкту

Скорочення витрат на персонал: Припустимо, що компанія витрачає близько $10,000 щомісяця на роботу технічного персоналу, який забезпечує обслуговування серверів у фізичному режимі. Впровадження KVM-over-IP дозволить зменшити потребу в фізичній присутності персоналу на 30%, що дасть змогу зекономити $3,000 щомісяця.
Зниження витрат на транспорт та відрядження: У разі фізичного обслуговування серверів, персонал може витрачати час і кошти на поїздки до дата-центрів або офісів. Припустимо, що транспортні витрати становлять $2,000 на місяць. Після впровадження віддаленого адміністрування за допомогою KVM-over-IP ці витрати зменшаться на 70%, що дозволить зекономити $1,400 щомісяця.
Скорочення часу простоїв: У випадку аварії віддалене адміністрування дозволяє оперативно вирішувати проблеми, що зменшує час простоїв і потенційні збитки. Якщо середня втрата від простою сервера складає $500 на годину, а віддалене адміністрування дозволяє зменшити час простою на 5 годин на місяць, це призведе до економії $2,500 на місяць.
Зниження витрат на обладнання та обслуговування: Впровадження KVM-over-IP дозволяє зменшити потребу в додатковому обладнанні для локального адміністрування та скоротити витрати на його обслуговування. Якщо щомісячна економія на обслуговуванні складає $1,000, це також додає до загальної економії.
Загальна економія та додатковий дохід від впровадження: 6,000+3,000+100,000=109,000 доларів на рік
Період окупності: 50,000/109,000≈0.46 року

Підсумуємо всі наведені вище щомісячні заощадження:

Загальна щомісячна економія: 3,000 + 1,400 + 2,500 + 1,000 = 7,900 дол. на місяць

Річна економія 7,900×12=94,800 дол. на рік

Економічна ефективність від впровадження системи віддаленого адміністрування за допомогою KVM-over-IP-перемикачів може становити майже $94,800 на рік за рахунок зниження витрат на персонал, транспорт, час простоїв і обслуговування обладнання.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Алгоритм просування проєкту програмної реалізації системи віддаленого адміністрування серверів із використанням KVM-over-IP-перемикачів схематично подано на рисунку 7.3.

Цей алгоритм дозволить комплексно просувати проєкт на ринку, охоплюючи різні канали й аудиторії, для досягнення оптимальної результативності та залучення клієнтів.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації каналів збуту та шляхів реалізації проєкту системи віддаленого адміністрування серверів із використанням KVM-over-IP-перемикачів можна скористатись кількома напрямами (рисунок 7.4).

Ці заходи допоможуть зробити збут і реалізацію проєкту більш ефективними, залучити нових клієнтів і забезпечити постійну підтримку для наявних.

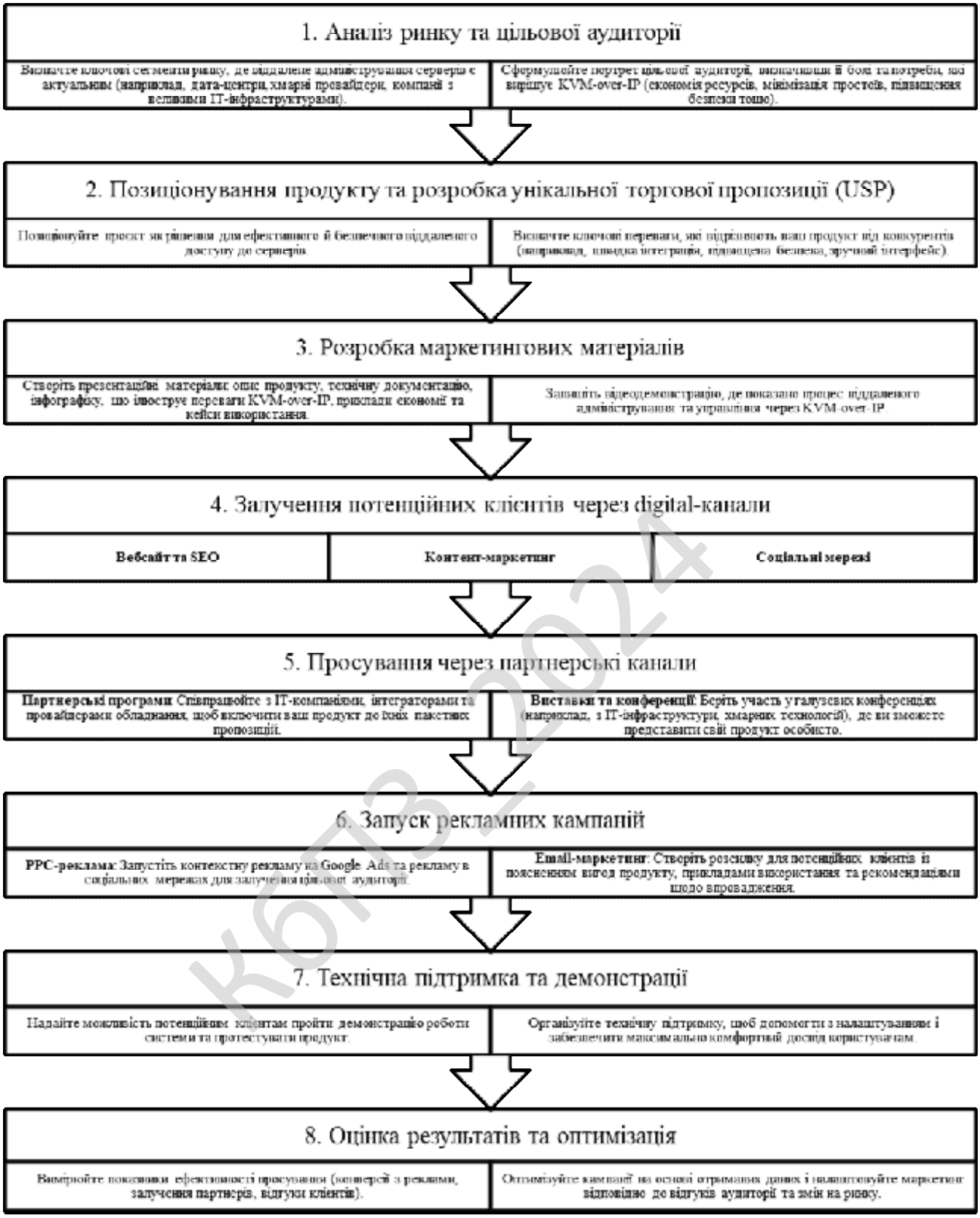

Рисунок 7.3 – Етапи просування проєкту

1. Аналіз каналів збуту та сегментація ринку

- **Визначте основні сегменти ринку**: IT-компанії, дата-центри, хмарні провайдери, середні та великі підприємства, що потребують віддаленого адміністрування серверів.
- **Сегментація клієнтів** дозволить адаптувати канали збуту для кожної категорії. Наприклад, дата-центри потребуватимуть високонадійного обладнання та швидкої підтримки, а менші компанії можуть зацікавитися базовими пакетами з доступом до ключових функцій.

2. Створення різних варіантів пакетів послуг

- **Базовий пакет**: Містить мінімально необхідний набір функцій за доступною ціною, що приверне увагу середніх підприємств і дозволить їм випробувати продукт.
- **Просунутий пакет**: Включає додаткові функції, наприклад, більше інструментів моніторингу та аналізу, які будуть цікаві для IT-компаній і хмарних провайдерів.
- **Корпоративний пакет**: Пропонує повний набір функцій, високу підтримку та можливість адаптації під вимоги конкретного клієнта. Він буде ідеальним для великих дата-центрів і корпоративних клієнтів.

3. Оптимізація партнерських каналів

- **Співпраця з постачальниками IT-обладнання**: Встановіть партнерські стосунки з постачальниками KVM-over-IP-перемикачів та іншого мережевого обладнання, які можуть запропонувати ваше рішення у складі своїх пакетів.
- **Аутсорсингові компанії та системні інтегратори**: Інтегратори можуть допомогти просувати рішення серед своїх клієнтів, для яких вони займаються проєктуванням та управлінням IT-інфраструктурою.

4. Використання цифрових каналів збуту

- **Власна веб-платформа**: Створіть простий у використанні сайт із можливістю отримати демо-версію продукту або консультацію.
- **IT-маркетплейси**: Пропонуйте програмний продукт на спеціалізованих маркетплейсах для IT-рішень, таких як AWS Marketplace, де велика аудиторія IT-фахівців і компаній може знайти ваш продукт.
- **Соціальні мережі та IT-спільноти**: LinkedIn, Reddit, та спеціалізовані форуми для IT-спеціалістів можуть бути ефективними каналами для залучення клієнтів через контент-маркетинг і обговорення продукту.

Рисунок 7.4 – Оптимізація каналів збуту та шляхів реалізації

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключовими факторами успіху проєкту програмної реалізації системи віддаленого адміністрування серверів за допомогою KVM-over-IP-перемикачів є представлені на рисунку 7.5 напрями.

Врахування цих факторів допоможе ефективно розробити та просунути проєкт, зробивши його конкурентоспроможним і привабливим для цільових клієнтів.

Рисунок 7.5 – Ключові фактори успіху проєкту

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Аналіз умов праці на робочому місці програміста

Загальна комп'ютеризація суспільства призвела до того, що використання комп'ютерів стало повсюдним у всіх сферах економіки та народного господарства. Застосування персональних комп'ютерів і ЕОМ дозволило значно підвищити продуктивність праці, змінити характер і зміст праці [4]. Комп'ютеризація, поряд з незаперечними перевагами, тягне за собою і багато проблем. Для того, щоб активне застосування комп'ютерних технологій не стало додатковим чинником погіршення здоров'я, вкрай необхідно щоб робоче місце відповідало гігієнічним вимогам. Темою дипломного проекту є розробка та дослідження та реалізація програмного продукту, тому актуально буде розгляд умов праці програміста.

Робота програміста пов'язана з постійною роботою на ЕОМ, яка відбувається у кімнаті розмірами 5м×7,2м×2,8м. Одна з її більших стін має шість двостулкових вікон, розмірами 2,2м×1,8м, які виходять на північний схід. Вікна розташовані рівномірно по всій довжині стіни. Підлога в кімнаті покрита леноліумом, всі стіни пофарбовані світло оранжевого кольору до висоти 2,8м, а далі підвісна стеля. Уздовж стін розташовані комп'ютерні столи. На них розташовуються 2 персональні комп'ютери й інша оргтехніка (сканер принтери, телефони й ксерокс). Столи мають пластикове покриття. Габарити їхньої робочої поверхні 1250 мм×850 мм. Висота столів 750 мм. Висота стільців від рівня підлоги становить 430 мм.

Згідно НПАОП 0.00 – 1.28 – 10 «Правила охорони праці під час електронно-обчислювальних машин» площа повинна задовольняти умові – не менш 6 м^2 на одне робоче місце. Кратність повітрообміну в приміщенні вузла також регламентується ДСанПіН 3.3.2.007 – 98, вона повинна становити 20

$м^3$/годину на одне місце. Виконання даних вимог забезпечить підтримку в приміщенні вузла оптимального значення вологості й складу повітря.

Відповідно ДБН В.2.5–28–2006 роботу програміста 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 вузла 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 сполученому висвітленні), повинен становити 0,5%, освітленість при штучному висвітленні повинна становити 300 лк.

За результатами виміру освітленості відділом охорони праці величина освітленості від системи загального штучного висвітлення лежить у межах 200-250 лк, що не відповідає вимогам, які пред'являються до приміщення.

Відповідно ДСанПіН 3.3.2.007 – 98 рівні звукового тиску в робочому приміщенні не повинні перевищувати в октавних смугах із середньо геометричними частотами наступних значень, наведених у таблиці 8.1.

У приміщенні перебувають наступні джерела шуму: електродвигуни внутрішнього вентилятора ЕОМ; працюючі принтери; працюючі дисководи. Шум, вироблений вентилятором можна класифікувати як постійний, всі інші джерела шуму, як імпульсні. Відповідно паспорта на приміщення рівень звуку, дБА, обмірюваний за шкалою (А) шумоміра досяг величини 28,3 дБА при роботі всього встаткування вузла, включаючи й ксерокс. Це дозволяє зробити висновок про відповідність рівня звуку в приміщенні вимогам нормативних актів.

Ергономічні вимоги до робочого місця працюючого з ВДТ ЕОМ і ПЕОМ нормуються НПАОП 0.00 – 1.28 – 10**.** Оптимальне положення тіла того, що працює забезпечується відповідною конструкцією робочого місця, а також регуляцією висоти робочої поверхні, сидіння, простори й підставки для ніг.

Даного місця програміста не мають регульованих параметрів. Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту дані в таблиці 8.2.

Таблиця 8.1 – Допустимі спектри рівнів звукового тиску

Робоче місце	Рівень звукового тиску, дБ, в октавних смугах із середньогеометричними частотами, Гц								Рівень шуму і еквівалентний рівень шуму, дБА
	63	125	250	500	1000	2000	4000	8000	
Приміщення конструкторських бюро, програмістів обчислювальних машин, лабораторій для теоретичних робіт	71	61	54	49	45	42	40	38	50

Таблиця 8.2 – Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту

Ріст людини, см	Висота робочої поверхні мм,	Висота простору для ніг, мм	Висота робочого сидіння, мм
175	765(740)	655(600)	450(440)

У дужках зазначені реальні значення параметрів робочого місця; всі вони не відповідають параметрам, зазначеним у стандарті.

Параметри мікроклімату можуть мінятися в широких межах, тоді як необхідною умовою життєдіяльності людини є підтримка сталості температури

тіла завдяки властивості терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах ДСН 3.3.6.042 – 99 встановлені величини параметрів мікроклімату, що створюють комфортні умови [1]. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. табл. 8.3).

Таблиця 8.3 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні	22 – 24°С
	Відносна вологість	40 – 60%
	Швидкість руху повітря	до 0,1 м/с
Теплий	Температура повітря в приміщенні	23 – 25°С
	Відносна вологість	40 ... 60%
	Швидкість руху повітря	0,1 ... 0,2 м/с

8.2 Розробка заходів з умов поліпшення охорони праці

Перерахуємо проведені заходи щодо забезпечення умов праці на робочому місці програміста.

З точки зору забезпечення електробезпеки до цих заходів можна віднести: устаткування розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв; періодична перевірка всіх приладів і пристроїв; щорічна здача іспитів з охорони праці.

З точки зору забезпечення оптимальних умов мікроклімату, рівня звуку і освітленості до цих заходів можна віднести: організацію природної вентиляції, за допомогою дефлектора, для забезпечення необхідного повітрообміну в

приміщенні вузла; організацію системи центрального опалювання, для підтримки оптимальної температури в холодний період року; організацію штучного загального освітлення, для забезпечення необхідних умов зорової роботи, що відповідають, оформлення паспорта на приміщення вузла, з занесенням в нього вимірювань освітленості і рівня звуку, проведених відділом охорони праці.

З точки зору забезпечення пожежної безпеки до цих заходів можна віднести наявність схеми евакуації з приміщення вузла, у випадку пожежі, повішену на вхідні двері.

Аналіз умов праці на робочому місці інженера-програміста показав, що на робочому місці не виконуються вимоги ергономіки. Для виконання їх можна запропонувати заміну не регульованого сидіння на крісло з регульованими ергономічними параметрами, а також заміну використовуваного столу на робоче місце оператора ЕОМ.

Різними фірмами в сукупності розроблено понад 11 схем регулювань параметрів робочого крісла, які забезпечують: плавне переміщення сидіння по висоті за допомогою газової пружини; плавна зміна нахилу спинки і сидіння; регулювання пружинного противотиску спинки крісла на спину оператора; перестановку спинки по висоті; зміна глибини сидіння шляхом зміни вигину краю сидіння; синхронне повторення рухів оператора сидінням і спинкою в правильному кутовому співвідношенні; синхронне повторення спинкою крісла рухів верхньої частини тулуба того, що сидить; амортизацію сидіння [4].

Найбільш популярними моделями комп'ютерних крісел, які володіють чотирма основними регулюваннями (висоти сидіння, висоти, глибини і нахилу спинки), є італійські, фінські моделі «Senior», «Volos», «Ergo», «Toronto», «Bini», «Metro», «NewStar», «Capris», «Fenix», «Xenus», «Quintus» і тому подібні. Крісла такої конструкції, відрегульовані відповідно до зростання і ваги оператора, а також характеру виконуваної роботи, дозволяють знизити навантаження на опорно-руховий апарат людини, що працює за комп'ютером.

8.3 Дослідження інформаційного навантаження на програміста

Програміст, у залежності від підготовки і досвіду, вирішує задачі різної складності, але в загальному випадку робота програміста будується по алгоритму, представленому у таблиці 8.4.

Таблиця 8.4 – Загальний алгоритм виконання завдань програмістом

Етап	Зміст	Витрата часу, %
I	Постановка задачі.	6.25
II	Вивчення матеріалу за поставленою задачею	
III	Визначення методу рішення задачі	6.25
IV	Складання алгоритму рішення задачі	12.5
V	Програмування	25
VI	Налагодження програми, складання звіту	50

Даний алгоритм відображає загальні дії програміста при рішенні поставленої задачі незалежно від її складності.

Підрахуємо кількість членів алгоритму і їхню частоту (імовірність) щодо загального числа, прийнятого за одиницю. Імовірність повторення і-ої ситуації визначається по формулі:

$$P_i = k/n,$$

де k – кількість повторень кожного елемента одного типу, n – сумарна кількість повторень від джерела інформації, одного типу.

Результати розрахунку зведемо в таблицю 8.6.

Таблиця 8.5 – Робота програміста

Етап	Шаг алгоритму	Зміст роботи	Літерне позначенн
1	2	3	4
I	1	Одержання першого варіанта технічного завдання	A
	2	Складання і уточнення технічного завдання	Ri
	3	Одержання остаточного варіанта технічного завдання	Сіл
	4	Складання переліку матеріалів, що	HII2
II	5	Вивчення матеріалів по тематиці задачі	Аг
	6	Вибір метолу рішення	C2J3
	7	Уточнення й узгодження обраного методу	B2
III	8	Остаточний вибір методу рішення	СчТ4
	9	Аналіз вхідної і вихідної інформації	B2
	10	Вибір мови програмування	C4TS
	11	Визначення структури програми	H3C5q
IV	12	Складання блок-схеми програми	C6q2
V	13	Логічний аналіз програми і коректування її	F1H4W2
	14	Компіляція програми	F2
	15	Виправлення помилок	D1W2
VI	16	Редагування програми	F2H5B3W
	17	Виконання програми	F3
	18	Аналіз результатів виконання	H6W5
	19	Тестування	C7W6
	20	Підготовка звіту про роботу	F4

Таблиця 8.6 – Результати розрахунку

Джерело	Етапи алгоритму	Символ	Кількість	Частота повторень
1	Аферентні – усього		6	1,00
	Вивчення технічної	А	2	0,33
	Спостереження	Р	4	0,67
2	Еферентні – усього		18	1,00
	Уточнення	В	3	0,17
	Вибір найкращого	С	8	0,44
	Виправлення	0	1	0.06
	Аналіз отриманих	н	6	0,33
	Виконання	к	0	0
	Логічні умови		13	1,00
3	Прийняття рішень на основі вивчення	І	5	0,39
	Грабічні матеріали	Ч	2	0.15
	Отриманого тексту	V	6	0.46
	Усього:		37	

Кількісні характеристики (Табл. 8.6) дозволяють розрахувати інформаційне навантаження програміста [8]. Ентропія інформації елементів кожного джерела інформації розраховується по формулі, біт/сигн:

$$H_j = -\sum_{i=1}^{m} p_i \log_2 p_i \quad \text{біт/сигн}$$

де m – число однотипних членів алгоритму розглянутого джерела інформації.

$$H_1 = 2 \times 2 + 2 \times 4 = 12 \text{ біт/сигн},$$

$$H_2 = 3 \times 1{,}585 + 8 \times 3 + 0 + 6 \times 2{,}585 = 44{,}265 \text{ біт/сигн},$$

$$H_3 = 5 \times 2{,}323 + 2 \times 1 + 6 + 2{,}585 = 29{,}125 \text{ біт/сигн}.$$

Потім визначається загальна ентропія інформації, біт/сигн:

$$H_s = H_1 + H_2 + H_3,$$

де H_I, H_2, H_3 – ентропія аферентних, еферентних елементів і логічних умов відповідно.

$$H_s = 10 + 44{,}265 + 29{,}125 = 83{,}39.$$

Далі визначається потік інформаційного навантаження біт/хв,

$$\Phi = \frac{H \cdot N}{t}$$

де N – сумарне число всіх членів алгоритму;

t – тривалість виконання всієї роботи, хв.

Від кожного джерела в інформації (члена алгоритму) у середньому надходить 3 інформаційних сигнали в годину, час роботи – 225 годин,

$$\Phi = \frac{83{,}39 \cdot 37 \cdot 3 \cdot 225}{13500} = 2{,}6 \text{ біт/хв},$$

Розраховане інформаційне навантаження порівнюється з припустимою. При необхідності приймається рішення про зміни в трудовому процесі.

Умови нормальної роботи виконуються при дотриманні співвідношення:

$$\Phi_{\text{доп.мін}} < \Phi_{\text{розр}} < \Phi_{\text{доп.макс}}$$

де $\Phi_{\text{доп.мін}}$, і $\Phi_{\text{доп.макс}}$ мінімальний і максимальний припустимі рівні інформаційних навантажень (0,8 і 3,2 біт/с відповідно);

$\Phi_{\text{розр}}$ – розрахункове інформаційне навантаження 0,8 < 2,6 <3,2 відповідає нормі.

8.4 Висновки до розділу

У цій частині магістерської роботи були розглянуті вплив факторів трудового і виробничого середовища програмістів, дослідження інформаційного навантаження на програміста. Дотримання умов, що визначають оптимальну організацію робочого місця програміста і навантаження, отримані ним в процесі роботи, дозволить зберегти гарну працездатність протягом усього робочого дня, підвищить, як у кількісному, так і в якісному відносинах продуктивність праці програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

– Досліджена система віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

– На основі отриманих результатів досліджень створена програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

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

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

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Twofish.

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

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Білан Я.Д. Дослідження та програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2024.

2. Оліфер В.Г. Комп'ютерні мережі. Принципи, технології, протоколи. Підручник / В.Г. Оліфер, Н.А.Оліфер. – [5-е вид.]. – 2016. – 944 с.

3. Е. Таненбаум, Д. Уезеролл «Комп'ютерні мережі». – [5-е вид.]. – 2016. – 960 с.

4. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

5. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

6. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

7. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

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

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

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

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

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

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

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

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

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

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

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

19. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In:

Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

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

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

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

23. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

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

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

26. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE*

International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

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

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

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

30. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

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

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

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

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

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

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

37. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

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

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

40. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного

процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

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

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

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

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

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

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Економічні вимоги.....5
8 Вимоги щодо охорони праці.....5
9 Перелік документів, що розробляються.....6
10 Етапи розробки.....6
11 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

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 віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів.

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 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 віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

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

7.2 Виконати маркетингове та економічне 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 повинно бути розглянуте дослідження інформаційного навантаження на програміста.

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 – 85 аркушів.

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 Маркетингове та економічне обґрунтування ІТ-проєкту.

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 18.12.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Петренюк В.І.

Дослідження та програмна реалізація системи віддаленого адміністрування серверами за допомогою KVM-over-IP-перемикачів

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 17

Літера: РП

Кропивницький – 2024 року

Основна програма

```
import socket
import threading
import json
import time
import logging

# Налаштування логування для зручності відстеження помилок
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -
%(message)s')

class KVMController:
    def __init__(self, ip, port, username, password):
        self.ip = ip
        self.port = port
        self.username = username
        self.password = password
        self.connection = None

    def connect(self):
        # Встановлення з'єднання з KVM-перемикачем
        try:
            self.connection = socket.create_connection((self.ip, self.port))
            self.authenticate()
        except Exception as e:
            logging.error(f"Connection failed: {e}")

    def authenticate(self):
        # Аутентифікація на KVM-перемикачі
        auth_data = json.dumps({"username": self.username, "password":
self.password})
        self.connection.send(auth_data.encode())
        response = self.connection.recv(1024).decode()
        if response != "authenticated":
            logging.error("Authentication failed")
            self.disconnect()

    def send_command(self, command):
        # Надсилання команди на KVM-перемикач
        try:
            self.connection.send(command.encode())
            response = self.connection.recv(1024)
            return response.decode()
        except Exception as e:
            logging.error(f"Error sending command: {e}")

    def get_status(self):
        # Отримання статусу KVM-перемикача
        return self.send_command("status")

    def reboot(self):
        # Перезавантаження сервера через KVM-перемикач
        return self.send_command("reboot")

    def disconnect(self):
        # Закриття з'єднання з KVM-перемикачем
        if self.connection:
            self.connection.close()
            self.connection = None


class Server:
    def __init__(self, host, port):
        self.host = host
        self.port = port
        self.clients = []
```

```
        self.kvm_controllers = {}

    def start(self):
        # Запуск серверної частини
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_socket.bind((self.host, self.port))
        server_socket.listen(5)
        logging.info("Server started and listening for connections")

        while True:
            client_socket, addr = server_socket.accept()
            logging.info(f"Connection from {addr}")
            client_thread = threading.Thread(target=self.handle_client,
args=(client_socket,))
            client_thread.start()

    def handle_client(self, client_socket):
        # Обробка підключення клієнта
        while True:
            try:
                data = client_socket.recv(1024).decode()
                if not data:
                    break
                response = self.process_command(data)
                client_socket.send(response.encode())
            except Exception as e:
                logging.error(f"Error handling client: {e}")
                break

        client_socket.close()

    def process_command(self, command_data):
        # Обробка команди, отриманої від клієнта
        try:
            command = json.loads(command_data)
            action = command.get("action")
            params = command.get("params", {})

            if action == "add_kvm":
                return self.add_kvm(params)
            elif action == "remove_kvm":
                return self.remove_kvm(params)
            elif action == "execute":
                return self.execute_command(params)
            else:
                return "Unknown action"
        except json.JSONDecodeError:
            return "Invalid command format"

    def add_kvm(self, params):
        # Додавання нового KVM-перемикача
        ip = params.get("ip")
        port = params.get("port")
        username = params.get("username")
        password = params.get("password")

        if not (ip and port and username and password):
            return "Missing KVM parameters"

        kvm = KVMController(ip, port, username, password)
        self.kvm_controllers[ip] = kvm
        kvm.connect()
        return f"KVM {ip} added and connected"

    def remove_kvm(self, params):
        # Видалення KVM-перемикача
        ip = params.get("ip")
        if ip in self.kvm_controllers:
            self.kvm_controllers[ip].disconnect()
```

```
            del self.kvm_controllers[ip]
            return f"KVM {ip} removed"
        else:
            return "KVM not found"

    def execute_command(self, params):
        # Виконання команди на KVM-перемикачі
        ip = params.get("ip")
        command = params.get("command")

        if ip not in self.kvm_controllers:
            return "KVM not found"

        kvm = self.kvm_controllers[ip]
        response = kvm.send_command(command)
        return response if response else "Command execution failed"


class Client:
    def __init__(self, server_ip, server_port):
        self.server_ip = server_ip
        self.server_port = server_port
        self.connection = None

    def connect(self):
        # Встановлення з'єднання з сервером
        try:
            self.connection = socket.create_connection((self.server_ip,
self.server_port))
        except Exception as e:
            logging.error(f"Connection to server failed: {e}")

    def send_command(self, command, params):
        # Надсилання команди на сервер
        command_data = json.dumps({"action": command, "params": params})
        try:
            self.connection.send(command_data.encode())
            response = self.connection.recv(1024).decode()
            return response
        except Exception as e:
            logging.error(f"Error sending command: {e}")

    def disconnect(self):
        # Закриття з'єднання з сервером
        if self.connection:
            self.connection.close()
            self.connection = None


if __name__ == "__main__":
    # Створення та запуск сервера
    server = Server("0.0.0.0", 9090)
    threading.Thread(target=server.start).start()

    # Затримка для запуску сервера
    time.sleep(2)

    # Приклад використання клієнта
    client = Client("127.0.0.1", 9090)
    client.connect()

    # Додавання нового KVM-перемикача
    print(client.send_command("add_kvm", {
        "ip": "192.168.0.100",
        "port": 5900,
        "username": "admin",
        "password": "password123"
    }))
```

```
# Виконання команди на KVM-перемикачі
print(client.send_command("execute", {
    "ip": "192.168.0.100",
    "command": "status"
}))

# Видалення KVM-перемикача
print(client.send_command("remove_kvm", {
    "ip": "192.168.0.100"
}))

client.disconnect()
```

Файл security.py

```
import ssl
import socket
import hashlib
import json

def create_encrypted_connection(ip, port):
    # Створення SSL-контексту для шифрування з'єднання
    context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
    context.check_hostname = False
    context.verify_mode = ssl.CERT_NONE

    encrypted_socket = context.wrap_socket(socket.socket(socket.AF_INET,
socket.SOCK_STREAM), server_hostname=ip)
    encrypted_socket.connect((ip, port))
    return encrypted_socket

def generate_token(username, password):
    # Генерація токену на основі користувача та паролю
    data = f"{username}:{password}"
    token = hashlib.sha256(data.encode()).hexdigest()
    return token

def validate_token(token, stored_token):
    # Перевірка токену
    return token == stored_token

def access_control_check(ip, suspicious_ips):
    # Механізм блокування або обмеження доступу при підозрілих діях
    if ip in suspicious_ips:
        return False
    return True

def integrate_with_ad(username, password):
    # Інтеграція з Active Directory
    # Емуляція перевірки через AD
    return username == "admin" and password == "password123"

# Приклад використання
if __name__ == "__main__":
    encrypted_conn = create_encrypted_connection("192.168.0.1", 443)
    token = generate_token("admin", "password123")
    print("Token validation:", validate_token(token, token))
    print("Access control check:", access_control_check("192.168.0.2",
["192.168.0.100"]))
```

Файл logging_and_backup.py

```
import sqlite3
import os
import datetime

def initialize_logging_db():
    # Ініціалізація бази даних для зберігання логів
    conn = sqlite3.connect('logs.db')
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS logs
                      (id INTEGER PRIMARY KEY, action TEXT, timestamp DATETIME
DEFAULT CURRENT_TIMESTAMP)''')
    conn.commit()
    conn.close()

def log_action(action):
    # Логування дії у базу даних
    conn = sqlite3.connect('logs.db')
    cursor = conn.cursor()
    cursor.execute("INSERT INTO logs (action) VALUES (?)", (action,))
    conn.commit()
    conn.close()

def create_backup():
    # Створення резервної копії бази даних
    backup_file =
f"backup_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}.db"
    if os.path.exists('logs.db'):
        os.system(f"cp logs.db {backup_file}")
        return backup_file
    return None

def log_detailed(action, level="INFO"):
    # Розширене логування з підтримкою рівнів деталізації
    log_action(f"[{level}] {action}")

# Приклад використання
if __name__ == "__main__":
    initialize_logging_db()
    log_action("Server started")
    backup_file = create_backup()
    log_detailed("Detailed log entry", "DEBUG")
    print(f"Backup created: {backup_file}")
```

Файл monitoring_and_discovery.py

```
import requests
import socket

def send_status_to_monitoring_system(server_id, status):
    # Надсилання статусу до моніторингової системи
    url = f"http://monitoring-system.local/update_status/{server_id}"
    data = {"status": status}
    response = requests.post(url, json=data)
    return response.status_code

def discover_kvm_devices(network_range):
    # Автоматичне виявлення KVM-перемикачів у мережі
    discovered_devices = []
    for ip in network_range:
        try:
            sock = socket.create_connection((ip, 5900), timeout=2)
            sock.close()
            discovered_devices.append(ip)
        except:
            pass
    return discovered_devices

def optimize_device_performance(devices):
    # Оптимізація роботи з великою кількістю KVM-перемикачів
    for device in devices:
        print(f"Optimizing {device}...")
        # Додаткова логіка оптимізації
    return True

# Приклад використання
if __name__ == "__main__":
    network_range = ["192.168.0." + str(i) for i in range(1, 255)]
    devices = discover_kvm_devices(network_range)
    print("Discovered KVM devices:", devices)
    optimize_device_performance(devices)
```

Файл task_automation.py

```
import subprocess

def automate_task(command):
    # Автоматизація виконання команди
    try:
        result = subprocess.run(command, shell=True, capture_output=True,
text=True)
        return result.stdout
    except Exception as e:
        return str(e)

def integrate_with_containers(container_name, action):
    # Інтеграція з контейнерами
    result = automate_task(f"docker {action} {container_name}")
    return result

def cli_support(command):
    # Підтримка командного інтерфейсу
    return automate_task(command)

# Приклад використання
if __name__ == "__main__":
    output = automate_task("echo 'Automation task running'")
    print(output)
    container_result = integrate_with_containers("my_container", "start")
    print(container_result)
```

Файл scalability_and_network_management.py

```
import threading
import time

def scale_server_instances(count):
    # Масштабування серверних інстансів
    for i in range(count):
        threading.Thread(target=run_server_instance, args=(i,)).start()

def run_server_instance(instance_id):
    # Запуск серверного інстансу
    time.sleep(1)  # Емуляція затримки запуску
    print(f"Server instance {instance_id} started")

def setup_redundant_channel(primary, backup):
    # Налаштування резервного каналу зв'язку
    print(f"Setting up redundant channel between {primary} and {backup}")

def manage_network():
    # Управління мережею
    print("Managing network configuration...")

# Приклад використання
if __name__ == "__main__":
    scale_server_instances(5)
    setup_redundant_channel("PrimaryServer", "BackupServer")
    manage_network()
```

Файл virtualization_and_configuration.py

```
def configure_server(ip, config):
    # Автоматична конфігурація сервера
    print(f"Configuring server {ip} with settings: {config}")

def support_virtual_machines(vm_name, action):
    # Підтримка віртуальних машин
    print(f"Performing {action} on virtual machine {vm_name}")

def auto_configure_kvm(kvm_ip, settings):
    # Автоматична конфігурація KVM-перемикача
    print(f"Auto-configuring KVM {kvm_ip} with settings: {settings}")

# Приклад використання
if __name__ == "__main__":
    configure_server("192.168.1.10", {"CPU": "4 cores", "RAM": "8GB"})
    support_virtual_machines("VM1", "start")
    auto_configure_kvm("192.168.1.50", {"resolution": "1920x1080"})
```

Файл multilanguage_support.py

```
import json

class LanguageSupport:
    def __init__(self, lang="en"):
        self.lang = lang
        self.translations = self.load_translations()

    def load_translations(self):
        # Завантаження файлу з перекладами залежно від обраної мови
        try:
            with open(f"translations/{self.lang}.json", "r") as file:
                return json.load(file)
        except FileNotFoundError:
            print("Translation file not found, loading default (English).")
            with open("translations/en.json", "r") as file:
                return json.load(file)

    def translate(self, key):
        # Отримання перекладу по ключу
        return self.translations.get(key, key)

# Приклад використання
if __name__ == "__main__":
    lang_support = LanguageSupport("uk")
    print(lang_support.translate("welcome_message"))
```

Файл devops_integration.py

```
import subprocess

def run_ansible_playbook(playbook_path, inventory_file):
    # Запуск плейбуку Ansible для автоматизації серверної інфраструктури
    command = f"ansible-playbook -i {inventory_file} {playbook_path}"
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    return result.stdout

def run_terraform_command(directory, action):
    # Запуск команди Terraform для управління інфраструктурою
    command = f"terraform -chdir={directory} {action}"
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    return result.stdout

# Приклад використання
if __name__ == "__main__":
    ansible_result = run_ansible_playbook("deploy.yml", "inventory.ini")
    print(ansible_result)

    terraform_result = run_terraform_command("/path/to/terraform", "apply")
    print(terraform_result)
```

Файл audit_module.py

```
import sqlite3
import datetime

def initialize_audit_db():
    # Ініціалізація бази даних для аудиту
    conn = sqlite3.connect('audit.db')
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS audit_log
                      (id INTEGER PRIMARY KEY, action TEXT, user TEXT, timestamp
DATETIME DEFAULT CURRENT_TIMESTAMP)''')
    conn.commit()
    conn.close()

def log_audit_action(action, user):
    # Логування дії у базу даних аудиту
    conn = sqlite3.connect('audit.db')
    cursor = conn.cursor()
    cursor.execute("INSERT INTO audit_log (action, user) VALUES (?, ?)",
(action, user))
    conn.commit()
    conn.close()

def get_audit_log():
    # Отримання логів аудиту
    conn = sqlite3.connect('audit.db')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM audit_log")
    rows = cursor.fetchall()
    conn.close()
    return rows

# Приклад використання
if __name__ == "__main__":
    initialize_audit_db()
    log_audit_action("User login", "admin")
    log = get_audit_log()
    print("Audit log:", log)
```

Файл analytics_module.py

```
import psutil
import time

def collect_cpu_usage(duration=10):
    # Збір даних про використання CPU протягом вказаного періоду
    cpu_usage = []
    for _ in range(duration):
        usage = psutil.cpu_percent(interval=1)
        cpu_usage.append(usage)
    return cpu_usage

def analyze_performance(cpu_data):
    # Аналіз продуктивності на основі даних про використання CPU
    avg_usage = sum(cpu_data) / len(cpu_data)
    if avg_usage > 80:
        return "High CPU usage detected"
    return "CPU usage is normal"

# Приклад використання
if __name__ == "__main__":
    cpu_data = collect_cpu_usage(5)
    result = analyze_performance(cpu_data)
    print(result)
```

Файл security_check.py

```
import requests

def check_security_threats(ip):
    # Перевірка на наявність загроз за допомогою стороннього сервісу IDS/IPS
    url = f"http://ids-service.local/check/{ip}"
    response = requests.get(url)
    if response.status_code == 200:
        threats = response.json().get("threats", [])
        if threats:
            return f"Threats detected: {threats}"
        return "No threats detected"
    return "Failed to connect to IDS service"

# Приклад використання
if __name__ == "__main__":
    threat_check = check_security_threats("192.168.0.101")
    print(threat_check)
```

Файл mfa_support.py

```
import pyotp
import time

def generate_mfa_secret():
    # Генерація секрету для багатофакторної аутентифікації
    return pyotp.random_base32()

def generate_mfa_token(secret):
    # Генерація токену на основі секрету
    totp = pyotp.TOTP(secret)
    return totp.now()

def validate_mfa_token(secret, token):
    # Перевірка токену MFA
    totp = pyotp.TOTP(secret)
    return totp.verify(token)

# Приклад використання
if __name__ == "__main__":
    secret = generate_mfa_secret()
    print("MFA Secret:", secret)

    token = generate_mfa_token(secret)
    print("Generated MFA Token:", token)

    is_valid = validate_mfa_token(secret, token)
    print("Token validation result:", is_valid)
```