Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи контролю Інтернет шлюзів на базі ОС Ubuntu”

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

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

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

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

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

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

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

Кулакевичу Артему Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи контролю Інтернет шлюзів на базі ОС Ubuntu*

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

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи контролю Інтернет шлюзів на базі ОС Ubuntu*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

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

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Кулакевич А.А.
(прізвище та ініціали)

АНОТАЦІЯ

Кулакевич А.А. Програмне забезпечення системи контролю Інтернет шлюзів на базі ОС Ubuntu. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи контролю Інтернет шлюзів на базі ОС Ubuntu.

Метою розробки є програмне забезпечення системи контролю Інтернет шлюзів на базі ОС Ubuntu.

Результат роботи – програмна реалізація системи контролю Інтернет шлюзів на базі ОС Ubuntu.

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 Ubuntu.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, система контролю Інтернет шлюзів, Ubuntu

ABSTRACT

Kulakevych A.A. Software for the control system of Internet gateways based on the Ubuntu OS. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the control system of Internet gateways based on the Ubuntu OS.

The purpose of the development is the software for the control system of Internet gateways based on the Ubuntu OS.

The result of the work is the software implementation of the control system of Internet gateways based on the Ubuntu OS.

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

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

The program can be used on a PC with the OS Ubuntu.

The program was developed in the Python environment.

Keywords: computer engineering, control system of Internet gateways, Ubuntu

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

HTML (HyperText Murkup Language) – мова розмітки гіпертекстових документів;

СУБД – системи управління базами даних;

TCP/IP (Transport Control Protocol/Internet Protocol) – протокол роботи мережі інтернет;

ГМ – глобальні мережі;

ЛВМ – локальна віртуальна мережа;

ОД – оконечне обладнання даних;

ОС – операційна система;

ПК – персональний комп'ютер;

ОЗП – основний запам'ятовуючий пристрій;

URL – universal resource locator – локатор ресурсів інтернет;

PPTP – Point-to-Point-Tunneling Protocol – протокол створення захищеного каналу при доступі віддалених користувачів через публічні мережі;

ВСЗІП – віртуальна система захисту інформаційного потоку;

ВСПІП – віртуальні системи поділу інформаційних потоків;

ВДТ – відеодисплейний термінал;

ЦО – цивільна оборона.

ПЕОМ – персональна електронно обчислювальна машина;

PHP – створює HTML-сторінки з використанням спеціальних тегів, розпізнаваних аналізатором PHP;

WWW – world wide web – всесвітня мережа інтернет;

FTP – File Transport Protocol – протокол передачі файлів.

ВСТУП

Актуальність теми. Інтернет-шлюз являє собою апаратно-програмний комплекс для організації доступу до зовнішньої мережі (Інтернет) з локальної мережі. Це один з робочих інструментів системного адміністратора, що дозволяє йому контролювати облік трафіку й доступ співробітників у зовнішню мережу.

Інтернет-шлюз надає можливості розподілу доступу серед користувачів, обліку трафіку, обмеження доступу для окремих користувачів або груп користувачів до ресурсів в Інтернет. Шлюз може включати проксі-сервер, міжмережний екран, поштовий сервер, шейпер, антивірус і інші мережні утиліти.

Інтернет-шлюз – сертифікований шлюз, при покупці якого користувач одержує повний комплекс супутніх рішень – антивірусний захист, детектор атак, модуль захисту від витоку конфіденційної інформації DLP, проксі сервер, можливості всебічного контролю трафіку й користувачів мережі, повноцінний VPN-сервер.

Шкідлива програма виявляється на етапі прикордонної маршрутизації на рівні прикладних протоколів (при передачі файлів або відкритті інтернет-сторінок) і поштового трафіку. Аналізується й весь внутрішній трафік, що проходить через шлюз, попереджаючи поширення внутрішніх погроз.

Інтернет-шлюз може працювати на одній з локальних машин мережі, під управлінням системи віртуалізації або на окремому сервері. Він може встановлюватися як програмне забезпечення на машину з робочою операційною системою, або на порожній комп'ютер з повною установкою власної операційної системи.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи контролю Інтернет шлюзів на базі ОС Ubuntu.

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

– Огляд існуючих систем контролю Інтернет шлюзів на базі ОС Ubuntu.

– Дослідження системи контролю Інтернет шлюзів на базі ОС Ubuntu.

– Програмна реалізація системи контролю Інтернет шлюзів на базі ОС Ubuntu.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі контролю Інтернет шлюзів на базі ОС Ubuntu.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи контролю Інтернет шлюзів на базі ОС Ubuntu, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Після установки розгортання системи шлюзу починається з визначення зовнішніх і внутрішніх інтерфейсів. Внутрішні інтерфейси визначають зв'язок з комп'ютерами в локальній мережі підприємства, а зовнішні – установлення зв'язку із провайдерами інтернету або іншої зовнішньої мережі. У більшості випадків на зовнішніх інтерфейсу настроюється сервіс NAT для транслювання внутрішніх адрес у зовнішні. Так як внутрішні комп'ютери мережі не видні зовні, NAT також допомагає захистити мережу.

Після налаштування мережних інтерфейсів наступний крок адміністратора – підключення й налаштування основних сервісів. Найбільше часто використовувані з них – DNS, DHCP і проксі-сервера.

DNS дозволяє використовувати інтернет-шлюз як у якості кешуючого DNS-сервера, так і як сервер, відповідального за який-небудь домен.

Сервер DHCP дозволяє додавати в мережу нові комп'ютери без необхідності прописувати вручну ip-адреса для кожного.

Проксі-сервер дозволяє одержувати детальну статистику обігів користувачів за протоколом HTTP/HTTPS, блокувати доступ користувачів до певних URL і оптимізувати роботу в інтернеті за рахунок кешування запитів, що, у підсумку приводить до деякої економії на споживанні трафіку.

Нарешті, на третьому етапі налаштування системний адміністратор визначає політики доступу зсередини й зовні локальної мережі. Цим завідує служба міжмережного екрана. Звичайно, за замовчуванням політики настроюються таким чином, щоб доступ до зовнішніх інтерфейсів шлюзу був повністю закритий, за винятком загальнодоступних сервісів, таких як поштовий або веб-сервер. Політика доступу з локальної мережі в більшості випадків настроюється протилежним образом: дозволене все, що не заборонено явно для конкретного користувача або мережі в цілому.

1.2 Область застосування

Ubuntu – це вільно розповсюджувана безкоштовна операційна система на основі Linux, що створена й підтримується численним співтовариством користувачів, розроблювачів і волонтерів із усього миру.

Сучасний світ уже неможливо представити без комп'ютерних пристроїв, що допомагають нам практично у всіх аспектах нашої діяльності. Чим би ви особисто або ваш проект не займалися, вам не обійтися без комп'ютерів. Вони дозволяють нам залишатися завжди на зв'язку, створювати творчі й робочі проекти, учитися новим навичкам і знанням, одержувати й обробляти інформацію, ділитися з миром своїми ідеями.

Робота будь-якого комп'ютерного пристрою залежить від його апаратного й програмного забезпечення. Найважливішим компонентом останнього є операційна система, під керуванням якої працює пристрій. На справжній момент найбільше поширення мають три основні операційні системи: Windows від компанії Microsoft, MacOS від Apple і сімейство вільних ОС на базі Linux, серед яких самої популярною є Ubuntu.

Незалежно від завдань вашого проекту ви напевно застосовуєте у своїй діяльності комп'ютери. Використання операційної системи Ubuntu зробить вашу роботу зручною, швидкою й значно скоротить витрати на обслуговування й підтримку.

Крім цього, можна виділити наступні переваги використання Ubuntu:

– Це вільно розповсюджувана ОС, що ви можете зовсім легально використовувати. Вам не прийде побоюватися перевірки легальності вашої роботи з боку органів, що контролюють права інтелектуальної власності.

– Ubuntu зовсім безкоштовна. Ви зможете заощадити значну суму засобів при переході на безкоштовне ПЗ. Нові пристрої без передвстановленої системи коштують, як правило, значно дешевше. Всі відновлення операційної системи Ubuntu ви теж завжди будете одержувати безкоштовно.

– Це надійно. ОС Ubuntu заснована на платформі Linux, що має багаторічну історію й величезну армію розроблювачів. Ця система зовсім стабільна й оперативно одержує всі виправлення й удосконалення. В Ubuntu немає вірусів, так що антивірусна програма вам не знадобиться.

– Додаткове програмне забезпечення. Відразу після установки системи ви одержите цілий комплект програм, які допоможуть вам вирішувати практично будь-які варті перед вами завдання. У тому числі офісний пакет, браузер, музичний і відеопрогравач і багато чого іншого. Якщо вам необхідно додаткове ПЗ, то ви зможете його в кілька кличів установити із Центра програмного забезпечення. І теж зовсім безкоштовно.

– Ubuntu динамічно розвивається. Кожні півроку виходить нова версія операційної системи, а версії LTS, що виходять раз в 2 роки, мають повну безкоштовну підтримку протягом 5 років. У розробці проекту бере участь не тільки багатотисячне співтовариство незалежних розроблювачів, але й компанія Canonical, що забезпечує планомірне впровадження всіх нових функцій і технологій.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи контролю Інтернет шлюзів на базі ОС Ubuntu, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

ІКС 2.3.4 (Інтернет Контроль Сервер)

ІКС 2.3.4 (Інтернет Контроль Сервер) від компанії "А-Реал Консалтинг" – це багатофункціональний міжмережний екран і проксі-сервер. Являє собою програмний комплекс, в основі якого лежить операційна система FreeBSD 8.1. Використання операційної системи FreeBSD означає мінімальне споживання системних ресурсів, високу надійність, безпеку й швидкість роботи.

Четверте відновлення ІКС 2.3 "Drumba the Dramatic Droid":

– Додано команду для одержання списку провайдерів і мереж з використанням Jabber/ ICQ-бота.

– Додано графіків завантаженості VLAN інтерфейсів.

– Додано можливість роботи веб-пошти по https.

– Тепер користувачі можуть міняти свій пароль у веб-інтерфейсі.

– Для мережного інтерфейсу тепер можна задати швидкість і режим роботи.

– Додано можливість перенаправляти GRE трафік.

– Додано можливість видалення звітів старше 2 місяців.

– Різні компоненти системи оновлені до останніх версій.

Укажемо найбільш важливі зміни версії 4.4:

– Ядро системи оновлене до FreeBSD 8.3.

– Підтримка роботи з WiFi і 3G.

– IP-телефонія підтримкою протоколів SIP і IAX.

– Layer-7 фільтрація.

– Додано функцію NAT для перенапрямку портів і VPN-мережі.

– Додано можливість роботи з поштовим сервером по протоколах SMTPs, IMAPs і POP3s.

– Категорії трафіку SkyDNS.

– Підтримка протоколу L2TP для VPN-підключень.

– Підтримка VMware Tools.

– Інші функціональні можливості.

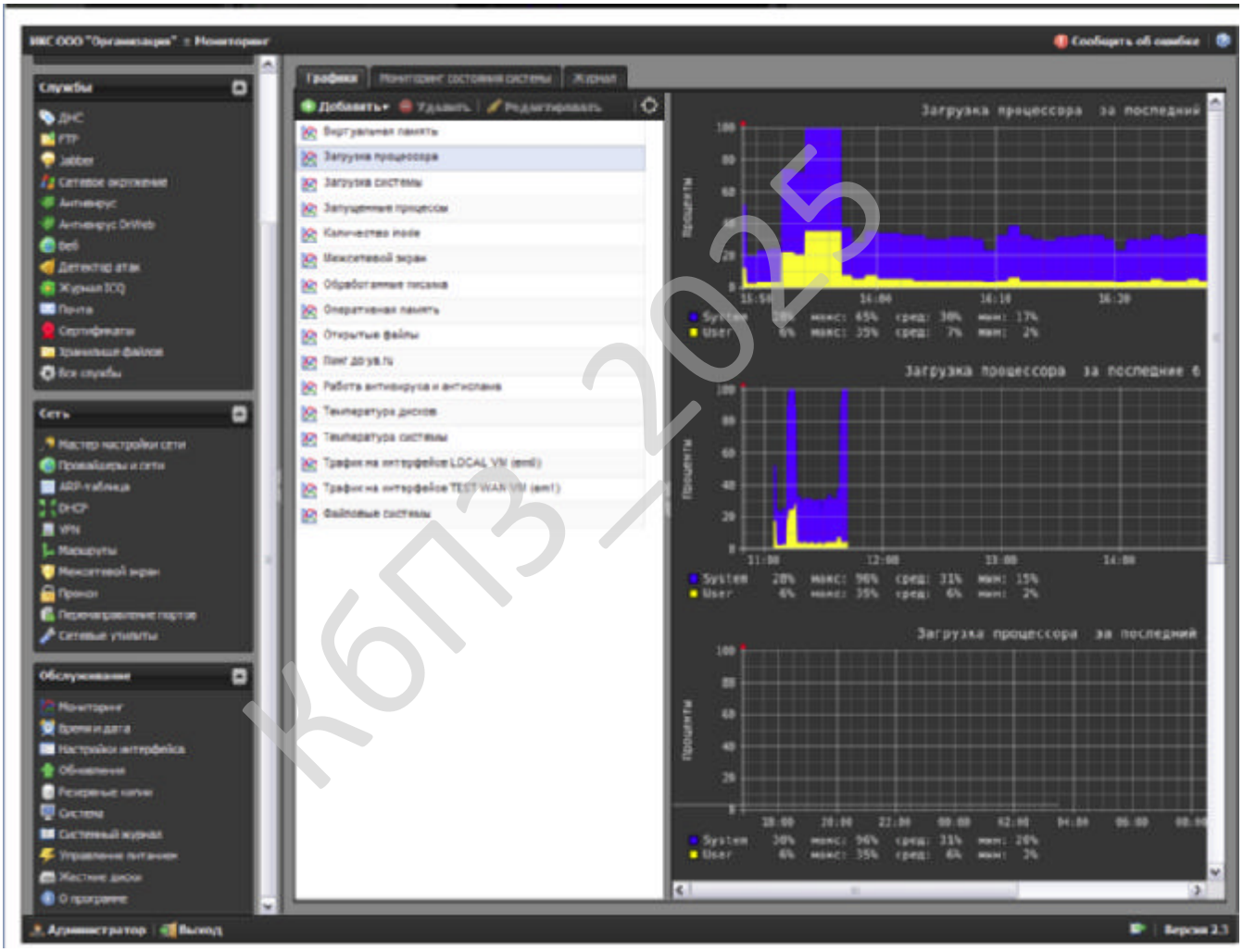

Рисунок 2.1 – Інтерфейс користувача міжмережного екрана ІКС

Установку міжмережного екрана ІКС можна проводити з компакт-диску або USB-накопичувача, попередньо записавши туди завантажений із сайту

виробника iso-образ, що містить програму установки міжмережного екрана ІКС. На даний момент відсутні iso-образи для віртуальних машин, які містять уже встановлений міжмережний екран ІКС. Сама установка гранично проста й проходить у кілька етапів. Програма-Установник має зовнішній вигляд подібний з sysinstall FreeBSD. На першому етапі установки буде запропонована вибрати мова інтерфейсу Після чого запуститься сама програма установки. Далі необхідно буде прийняти ліцензійну угоду, вибрати мережний інтерфейс, що підключений до локальної мережі, провести його налаштування й вибрати жорсткий диск для установки. Після підтвердження всіх заданих налаштувань і вибору жорсткого диска для установки, програма установки сама розіб'є й відформатує жорсткий диск, а потім установка продовжиться. По закінченні установки, буде показане повідомлення-нагадування які дані потрібно використовувати для підключення до веб-консолі міжмережного екрана ІКС.

Функціональність і робота з міжмережним екраном

Основним засобом керування міжмережним екраном є веб-інтерфейс. Сам веб-інтерфейс залишає приємне враження. Для полегшення навігації й доступу до потрібних сервісів і пунктів меню, використовується чотири логічних розділи: «Користувачі й статистика», «Служби», «Мережа» і «Обслуговування».

Розділ «Користувачі». У цьому розділі згруповані модулі керування користувачами, групами користувачів, їхніми ролями. А також модуль для роботи з наборами правил (у тому числі контентної фільтрації), модуль для керування категоріями даних і модуль відображення детальної статистики роботи користувачів у мережі Інтернет.

У розділі «Служби» відображається список запущених служб міжмережного екрана ІКС. Клацнувши на будь-якій службі, можна побачити вікно стану служби, а також одержати доступ до додаткових налаштувань цієї служби або зупинити її роботу.

Модуль «Всі служби» містить у собі список всіх мережних сервісів, доступних у міжмережномуу екрані ІКС.

Розділ «Мережа» містить у собі засобу керування мережними підключеннями, керування маршрутизацією, міжмережним екраном, системою виявлення вторгнень, проксі-сервером, а також DHCP і VPN серверами, які входять до складу міжмережного екрана ІКС.

А в розділі «Обслуговування» згруповані системні модулі керування міжмережним екраном ІКС, у тому числі й устаткуванням, на якому він працює.

Для приклада покажемо вікно стану однієї зі служб міжмережного екрана ІКС. У першу чергу необхідно провести налаштування мережних інтерфейсів, щоб забезпечити міжмережному екрану ІКС доступ до мережі Інтернет, а потім провести відновлення міжмережного екрана ІКС. Легше всього провести налаштування мережі за допомогою майстра налаштування мережі. Якщо необхідно провести більше тонке налаштування із вказівкою додаткових параметрів, то це можна зробити вручну. Майстер первісного налаштування мережі можна запустити з модуля Мережа, клацнувши по відповідний меню. Як бачимо, нам пропонують широкий вибір варіантів підключень.

Перед підтвердженням налаштувань мережних інтерфейсів буде показана підсумкова таблиця змін. Якщо якісь параметри задані невірно, можна повернутися на кілька кроків назад і підкорегувати налаштування.

Після того, як всі налаштування були підтверджені й застосовані, відбудеться автоматична переадресація в меню «Провайдери й мережі» модуля «Мережа». У цьому меню можна подивитися список і стан мережних інтерфейсів, відключити або включити ті або інші інтерфейси, провести їхнє налаштування. Також можна подивитися журнал роботи комплексу.

У правильності з'єднання з мережею Інтернет можна переконатися, скориставшись стандартною утилітою Ping. Вона доступна в меню «Мережні утиліти» модуля «Мережа». Якщо необхідно, можна зробити й трасування, використавши утиліту traceroute, що перебуває в тім же меню.

У цілому, налаштування мережних інтерфейсів труднощів не викликає. Після того, як ми змогли підключитися до мережі Інтернет, потрібно перевірити

наявність відновлення. Зробити це можна клацнувши на пункт меню «Відновлення» модуля «Обслуговування».

Як бачимо, які-небудь відновлення відсутні. На цьому коротке первісне налаштування ІКС можна вважати кінченої. Подальше налаштування міжмережного екрана ІКС залежать від конкретного сценарію використання.

Міжмережний екран ІКС залишив приємне враження. Це добре продуманий програмний продукт із дуже широкими можливостями. Його легко освоїти персоналу, що вперше зіштовхується з подібним класом продуктів або ж не мав достатнього досвіду роботи. Для досвідчених системних адміністраторів у міжмережному екрані ІКС передбачена можливість як завгодно тонкого налаштування параметрів, а також широкий вибір мережних сервісів для побудови системи високої складності. Що дає можливість найбільше повно врахувати вимоги тої або іншої організації в створенні мережної інфраструктури.

Переваги:

– Використання як основу операційної системи FreeBSD. Велика кількість мережних сервісів, у тому числі FTP, Web, DNS, DHCP, VPN, проксі, поштовий і jabber сервера, LDAP, і інші.

– Використання популярної IPS Snort. Можливість перевірки трафіку за допомогою антивірусів ClamAV, що є безкоштовним і Dr.Web, для роботи з яким потрібно придбати окрему ліцензію.

– Можливість побудови гнучкої системи правил обробки мережного трафіку, у тому числі можливість контентної фільтрації, пріоритезація трафіку, маршрутизація, перенапрямок портів, вказівка тимчасових діапазонів роботи правил обробки трафіку, керування смугою пропущення та інше.

– Підтримка роботи на віртуальних машинах. Але передналаштовані образи відсутні.

– Система моніторингу встаткування, компонентів міжмережного екрана ІКС і самої системи.

– Простота й наочність у налаштуванні, зручний веб-інтерфейс.

– Пробний період використання становить 35 днів.

– Безкоштовна технічна підтримка й гарна документація.

– Довічна ліцензія, не обмежена за часом використання.

Недоліки:

– Використання як основу операційної системи FreeBSD. Так, це як плюс, так і мінус. Мінус у плані підтримуваного встаткування. В операційної системи FreeBSD є певні проблеми в цьому напрямку. На жаль, операційна система FreeBSD небагато відстає, у плані підтримки нового обладнання, від Linux і, тим більше, від Microsoft Windows. Тому, ми рекомендуємо перед установкою міжмережного екрана ІКС переконатися, що встаткування підтримується операційною системою FreeBSD.

– Відсутність як провайдер Wi-Fi з'єднання. З урахуванням розвитку бездротових мереж – це актуально. Розроблювачі додали підтримку Wi-Fi з'єднань у версії ІКС 4.4.

– У процесі підготовки огляду були проблеми з використанням мережного адаптера PCnet FAST III у віртуальній машині Virtualbox. Система коректно заробила тільки з мережним адаптером Intel.

– Модуль «Відновлення» повідомляє про відсутність відновлень, незважаючи на те, що нове відновлення в списку є.

– Проблеми роботи з консоллю в деяких менш популярних браузерах. Наприклад, Internet Explorer 8 досить часто пропонував зупинити сценарій виконання на сторінці, при роботі з веб-інтерфейсом.

– Консоль відновлення досить обмежена в можливостях. Наприклад, бажано мати можливість потрапити в стандартну оболонку, для прямої роботи з операційною системою міжмережного екрана ІКС в особливо важких випадках.

– Хотілося б мати можливість додаткової установки пакетів програмного забезпечення, зі списку підтримуваних розроблювачами, з інтеграцією у веб-інтерфейс. Це розширило б можливості продукту в цілому.

Ideco ICS

Ключовим завданням, що вирішує Ideco ICS, є, звичайно ж, керування трафіком – від маршрутизації до шифрування й балансування навантаження. Будучи встановленим на границі корпоративної й глобальної мереж, шлюз дозволяє контролювати й при необхідності обмежувати доступ співробітників до онлайнових ресурсів і веб-контенту певного змісту. Передбачено й засобу обмеження швидкості й обсягу переданих користувачами даних. Ліміти можуть бути виставлені як для окремих співробітників компанії, так і по відділах і підприємству в цілому. Модуль формування звітів для директора й IT-менеджера допомагає оперативно оцінювати ступінь цільового використання мережних ресурсів офісними співробітниками.

Ideco ICS забезпечує захист локальної мережі від зовнішніх погроз і витоків конфіденційної інформації. Убудований у шлюз модуль Data Leak Prevention (DLP) сканує вихідний трафік і блокує передачу захищених документів через електронну пошту й веб-протоколи, розпізнаючи конфіденційні дані за допомогою технології цифрових відбитків. Перехоплені файли, що попадають у категорію "для службового користування", можуть бути збережені на сервері для подальшого їхнього перегляду адміністратором або співробітником служби безпеки.

До складу Ideco Internet Control Server включені також брандмауер, детектор мережних атак, антивірус і антиспам. Залежно від редакції й умов ліцензування, у продукті можуть бути задіяні антиспамові й антивірусні модулі різних виробників. Використання в шлюзі додаткових засобів захисту особливо актуально у світлі того, що вітчизняні організації SMB-сегмента не приділяють належну увагу питанням інформаційної безпеки.

Використовуючи технологію віртуальних приватних мереж VPN, шлюз Ideco ICS дозволяє організувати підключення вилучених користувачів по захищеному каналі до локальної мережі організації. За допомогою VPN можна налагодити взаємодія мобільних співробітників, а також об'єднати вилучені офіси

компанії в єдиний інформаційний простір. Результатом цього є підвищення ефективності бізнес-процесів для територіально розподілених підприємств.

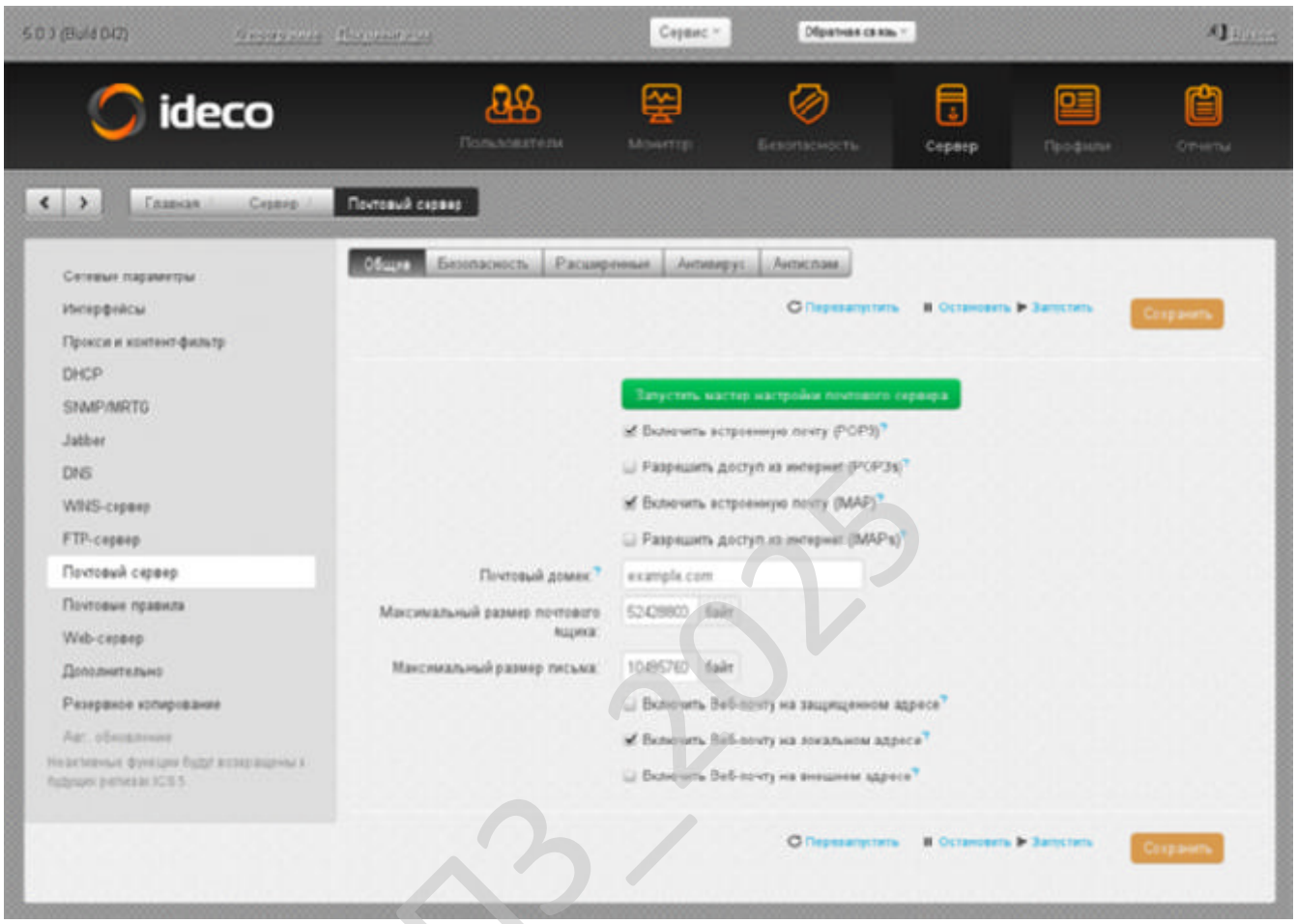

Рисунок 2.2. – інтерфейс користувача Ideco ICS

П'ята версія шлюзу Ideco ICS побудована на новій технологічній платформі й оснащена поліпшеним інтерфейсом адміністратора

Крім цього, в Ideco ICS інтегровані різні інтернет-сервіси. Зокрема, у дистрибутив продукту входить зконфігурований сервер електронної пошти, що містить у собі всі можливості, необхідні для створення комунікаційного середовища в малих і середніх компаніях. Керування email-сервером і робота з електронною кореспонденцією можуть здійснюватися через веб-інтерфейс із будь-якої точки миру, що забезпечує високий рівень мобільності співробітників.

З метою безпеки трансльований між клієнтом і поштовим сервером трафік шифрується. До складу шлюзу також включена підтримка обміну миттєвими повідомленнями в корпоративній мережі за протоколом Jabber. Це дозволяє виключити пересилання конфіденційної інформації через Інтернет, як це відбувається при використанні традиційних систем на зразок ICQ.

Окремого згадування заслуговує повноцінний веб-сервер, що дозволяє розвертати на базі Ideco ICS онлайнові площадки з підтримкою сучасних інтернет-технологій. На додаток до цього в продукті передбачений FTP-сервер для зберігання файлів, доступ до яких може здійснюватися користувачами як із внутрішньої, так і зовнішньої мережі.

Немаловажною особливістю шлюзу є підтримка Active Directory, що дозволяє в автоматичному режимі здійснювати перенос існуючої структури підприємства й налаштувань користувальницького робітничого середовища зі служби каталогів Windows в Ideco ICS. Для використання всіх можливостей інтеграції з Active Directory необхідно синхронізувати інтернет-шлюз із операційною системою Windows Server. Для цього досить створити групу в дереві користувачів, призначити параметри синхронізації й вибрати імпортовані об'єкти. Ideco ICS підтримує протокол мережний автентифікації NTLM (NT LAN Manager), що дозволяє будь-якому зареєструвались у домені користувачеві виходити в Інтернет без додаткової авторизації на шлюзі.

Такі основні функціональні можливості Ideco Internet Control Server, що роблять інтернет-шлюз універсальним засобом для керування IT-Інфраструктурою в компаніях і захисту локальної мережі від зовнішніх погроз. Програмний комплекс поширюється розроблювачем у декількох редакціях, серед яких представлені безкоштовні складання для домашніх користувачів і малих організацій (до п'яти співробітників).

Для установки й роботи Ideco ICS не потрібно передвстановлена операційна система й додаткове програмне забезпечення. Шлюз установлюється на виділений сервер із завантажувального накопичувача, при цьому автоматично

створюється файлова система, і розвертаються всі необхідні компоненти. Мінімальні вимоги до обчислювальної машини такі: процесор Intel із частотою не менш 1 Ггц, обсяг оперативної пам'яті не менш 1 Гбайт і два мережних адаптери. При розгортанні шлюзу розроблювачі рекомендують дотримуватися наступного правила: на кожні 250 активних користувачів потрібно один гігабайт оперативної пам'яті й одне ядро процесора. Вимоги можуть варіюватися залежно від мережного навантаження й використовуваних сервісів, таких як контентна фільтрація й антивіруси. Залежно від інтенсивності використання доступу в глобальну Мережу й тривалості зберігання деталізованої статистики може знадобитися установка додаткового дискового накопичувача. Для зберігання даних убудованих служб (FTP, веб-сервер, поштовий сервер, проксі-сервер і інших) також знадобиться додатковий простір.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це об’єктно-орієнтована мова програмування високого рівня загального призначення з відкритим кодом. Це визначення може бути важким для новачків, тому розглянемо кожну характеристику окремо, щоб зрозуміти, що вона означає:

– Відкритий вихідний код: це безкоштовно та доступно для подальших покращень, таких як додавання корисних функцій або виправлення помилок.

– Об’єктно-орієнтована: заснована не на функціях, але в об’єктах з певними атрибутами й методами.

– Високий рівень: зручний для людини, а не для комп’ютера.

– Загальне призначення: можна використовувати для створення будь-яких програм.

Ця мова використовується в будь-якому програмному забезпеченні, про яке ви тільки можете подумати. Ви можете використовувати його для створення

веб-сайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу та багато іншого. Також застосовується в науці про дані, аналізі даних, машинному навчанні, інженерії даних, веб-розробці, розробці програмного забезпечення та інших галузях.

Переваги та недоліки Python

Переваги:

– Її легко читати, вчити та писати. Це мова програмування високого рівня з англійським синтаксисом. Це полегшує читання та розуміння коду. Її дійсно легко зрозуміти і вивчити, тому багато людей рекомендують Python новачкам. Вам потрібно менше рядків коду для виконання того ж завдання в порівнянні з іншими основними мовами, такими як C/C++ та Java.

– Підвищує продуктивність. Це дуже продуктивна мова. Завдяки її простоті розробники можуть зосередитися на розв'язанні проблеми. Їм не потрібно витрачати багато часу на розуміння синтаксису або поведінку мови програмування. Ви пишете менше коду та виконуєте більше завдань.

– Інтерпретована мова. Python мова, що інтерпретується, а це означає, що вона безпосередньо виконує код по рядку. Якщо сталася помилка, вона зупиняє подальше виконання та повідомляє про її виникнення. Вона показує лише одну помилку, навіть якщо у програмі їх кілька. Це спрощує налагодження.

– Динамічно типізована. Python не визначає тип змінної, доки ми не запустимо код. Вона автоматично надає тип даних, коли відбувається процес виконання. Фахівець може не турбуватися про оголошення змінних та типи даних.

– Безкоштовна та з відкритим вихідним кодом. Ця мова постачається під схваленою OSI ліцензією з відкритим вихідним кодом. Це робить його безкоштовним для використання та розповсюдження. Ви можете завантажити вихідний код, змінити його та навіть розповсюджувати свою версію. Це корисно для організацій, які хочуть використати свою версію для розробки.

– Підтримка великих бібліотек. Стандартна бібліотека Python є величезною, ви можете знайти майже всі функції, необхідні для вашого завдання. Таким чином ви не залежите від зовнішніх бібліотек.

– Портативність .У багатьох мовах, таких як C/C++, потрібно змінити свій код, щоб запустити програму на різних платформах. З Python все інакше. Ви тільки пишете один раз і запускаєте її будь-де.

Недоліки:

– Низька швидкість. Вище ми обговорювали, що це інтерпретована мова з динамічною типізацією. Порядкове виконання коду часто призводить до повільного виконання. Динамічна природа Python також є причиною її низької швидкості, оскільки їй доводиться виконувати додаткову роботу при виконанні коду. Тому вона не підходить для цілей, де швидкість важливий аспект проєкту.

– Неефективна для пам’яті. Ця мова програмування використовує великий обсяг пам’яті, це може бути недоліком при створенні програм, коли віддають перевагу оптимізації пам’яті.

– Слабка у мобільних обчисленнях. Python зазвичай використовується у серверному програмуванні. Ми не бачимо – її на стороні клієнта або в мобільних програмах з таких причин: вона не заощаджує пам’ять і має повільну обчислювальну потужність у порівнянні з іншими мовами.

– Доступ до бази даних. Програмувати на цій мові легко, але коли ми взаємодіємо з базою даних, її не вистачає. Рівень доступу до бази даних у Python примітивний та недостатньо розвинений у порівнянні з іншими популярними технологіями.

– Помилки виконання. Це мова з динамічною типізацією, тому тип даних змінної може змінюватись у будь-який час. Змінна, що містить ціле число, у майбутньому може містити рядок, що може призвести до помилок виконання.

Застосування Python:

– Для аналізу даних. Дані стали цінним активом у будь-якій сучасній галузі, і більшість компаній зацікавлені у збиранні, обробці та аналізі

релевантних даних, щоб витягти з них цінну інформацію для бізнесу. І тут Python виходить за межі будь-якої конкуренції. Python особливо цінна тим, що крім великої стандартної бібліотеки надає величезний набір додаткових модулів, розроблених спеціально для аналітичних цілей. Найвідоміші бібліотеки Python для аналізу даних – це pandas і NumPy . Ці інструменти дозволяють робити з вашими даними майже все, наприклад, очищати і аналізувати їх, вивчати статистику або візуалізувати приховані тенденції у ваших даних.

– Для візуалізації даних. Візуалізація даних – це окрема частина аналізу даних, яка допомагає нам подавати інформацію, необроблену чи очищену, у більш змістовній формі. Тут Python знову входить у гру, пропонуючи широкий спектр інструментів візуалізації даних. Найпопулярніші з них – matplotlib і заснований на ній seaborn. Використовуючи їх, ми можемо створювати буквально всі види візуалізації: від найпростіших до складніших.

– Для машинного навчання. Машинне навчання (ML) є основою більшості завдань науки даних. Він є областю штучного інтелекту, пов’язаною з використанням алгоритмів, що дозволяють машинам вивчати закономірності та тенденції на основі історичних даних, щоб робити прогнози на основі невідомих даних. – Використовуючи методи ML, ми можемо створювати моделі, які можуть точно передбачити швидкість відтоку клієнтів компанії, оцінити ризик виникнення у людини певного захворювання, визначити оптимальне розташування автомобілів таксі й т.д. За допомогою Python ми можемо побудувати модель ML, використовуючи лише три рядки коду.

– Для розробки програмного забезпечення. Крім свого багатостороннього застосування в галузях науки про дані, Python використовується на кожному етапі розробки програмного забезпечення, включаючи контроль складання, автоматичну безперервну компіляцію, прототипування, відстеження помилок, тестування та обслуговування програмного забезпечення. За допомогою цієї мови можемо створювати аудіо- або відеопрограми на основі методів штучного інтелекту, машинного навчання, API (інтерфейсів прикладного програмування),

GUI (графічних інтерфейсів) або будь-якого іншого типу програмного забезпечення.

– Для веброзробки. У той час як для створення візуальної частини вебсайту ми переважно будемо використовувати такі мови, як HTML, CSS та JavaScript, для його невидимої частини ми часто вибираємо Python. Серед масштабних вебсайтів та програм, створених за допомогою цієї мови, варто згадати Google, Facebook, Instagram, YouTube, Dropbox та Reddit.

– Для автоматизації задач/скриптингу. Це відмінний інструмент для написання програм для автоматизації різних завдань, що повторюються. Цей процес називається скриптингом. Зокрема, можна робити скрипти для роботи з файлами та папками. Наприклад, можна створювати, перейменовувати, перетворювати, розділяти, об'єднувати або видаляти файли, перевіряти їх на наявність помилок. Ви також можете використовувати автоматизацію Python для пошуку та завантаження інформації з Інтернету, заповнення та надсилання онлайн-форм та надсилання регулярних повідомлень або електронних листів.

Яким фахівцям потрібно володіти Python:

– Фахівець з даних.

– Аналітик даних.

– Інженер даних.

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веб-розробника.

– Backend-розробник.

– DevOps-інженер.

– Інженер-програміст.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи контролю Інтернет шлюзів на базі ОС Ubuntu.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Ріст кількості локальних обчислювальних мереж і гостра необхідність надання доступу до мережі Інтернет все більшому числу пристроїв, а також захисту їх від атак ззовні, привели до розвитку програмних, апаратних і програмно-апаратних комплексів, за допомогою яких можна надати доступ до мережі Інтернет великій кількості пристроїв у локальній мережі. Ці комплекси мають широку функціональність, засобами захисту від зовнішніх погроз (міжмережні екрани, засоби виявлення вторгнень, антивірусна перевірка контента), модулі контентної фільтрації, засобу для збору статистичних даних, керування пропускною здатністю каналу та інше.

До ключових особливостей інтернет-шлюзу варто віднести:

– Велика кількість мережних сервісів, у тому числі FTP, Web, DNS, DHCP, VPN, проксі, поштовий і jabber сервера, LDAP, і інші. Це дозволяє будувати конфігурації системи практично довільного рівня складності, залежно від сценарію використання.

– Для доступу до мережі Інтернет можуть бути використані різні варіанти підключення до провайдерів. Підтримуються: DHCP; PPPOE; PPTP over IP; PPTP over DHCP; провайдер VLAN.

– Система виявлення вторгнень (IPS) Snort.

– Можливість перевірки трафіку антивірусами ClamAV (безкоштовний продукт) і Dr.Web (для роботи потрібне придбання окремої ліцензії).

– Проксування, у тому числі й прозоре.

– Модуль контентної фільтрації.

– Гнучка система роботи із правилами для користувачів, груп користувачів.

– Зручне налаштування ширини смуги пропущення.

– Інтеграція з мережами Microsoft Windows.

– Підтримка установки й роботи на віртуальних машинах Virtualbox, VmWare Workstation і VmWare ESXi.

– Інтернет шлюз сертифікований.

Інтернет шлюз має широку функціональність і містить безліч мережних сервісів. Так, Інтернет шлюз може виступати в ролі повноцінного:

– DHCP-сервера, для автоматичного налаштування мережних параметрів клієнтських пристроїв локальної мережі.

– FTP-сервера (Proftpd), для забезпечення доступу до файлів за протоколом FTP.

– WEB-сервера (Apache), з підтримкою віртуальних хостов, для розміщення сайтів і забезпечення до них доступу.

– Mail-сервера (Postfix), з можливостями збору пошти з інших поштових скриньок і убудованим спам-фільтром, для передачі й прийому електронної пошти.

– DNS-сервера (Bind), для роботи зі службами дозволу імен.

– Сервера каталогів, для синхронізації з мережами Microsoft Windows і Microsoft Active Directory.

– HTTP проксі-сервера (squid), з підтримкою прозорого проксірования, перевірки трафіку за допомогою антивірусів ClamAV і DrWeb, облік і фільтрація трафіку по URL і mime-типах, кешування сторінок (для економії трафіку), а також авторизації для доступу до проксі-сервера за обліковим даними користувачів Інтернет шлюзу або Active Directory.

– Socks5 проксі-сервера, для надання можливості прозорого використання сервісів за Інтернет шлюзом.

– VPN-сервера, для надання клієнтам вилученого доступу за допомогою VPN-авторизації.

– Сервера часу, для можливості синхронізації часу клієнтських пристроїв із сервером, що особливо актуально в мережах з Active Directory.

Як бачимо, Інтернет шлюз надає широкі можливості для використання різних мережних сервісів і побудови конфігурацій по-різному складності, залежно від потреб тої або іншої організації. До того ж, є можливість залучення сервісів у міру росту потреб, без використання додаткового стороннього програмного забезпечення й окремих серверів для цих сервісів. Так само, в Інтернет шлюзі, крім перерахованого вище, є ряд додаткових можливостей. Розглянемо і їх.

Додаткові функціональні модулі Інтернет шлюзу

«Журнал ICQ». Цей сервіс служить для перехоплення повідомлень ICQ і може використовуватися як простий модуль DLP-системи для контролю переписки співробітників і встановлення факту витоку конфіденційної інформації. «Журнал ICQ» працює не з усіма сторонніми клієнтами служби ICQ і не перехоплює зашифровані повідомлення.

Якщо провайдер надає динамічну IP-адресу, а доступ до сервера з мережі Інтернет необхідний, то можна скористатися модулем «DynDNS» (динамічний DNS). Попередньо потрібно зареєструватися на сайті http://dyndns.org/ або http://no-ip.com, і зареєструвати своє власне доменне ім'я. Після чого вказати облікові дані в модулі «DynDNS» для доступу до облікового запису, указати провайдеру й ip-адреса, що буде мінятися, і задати ім'я хосту. Зручна функція, якщо немає можливості одержати статичний ip-адресу, але, приміром, хочеться щоб свій власний сайт на своєму сервері був доступний у мережі Інтернет.

Якщо в організації використовуються апаратні маршрутизатори фірми Cisco, то Інтернет шлюз може підключатися до них за протоколом Netflow v.5 для збору й обробки статистичної інформації. Що може використовуватися для більше повного обліку обсягів трафіку, аналізу відвідуваних ресурсів та інше.

До складу Інтернет шлюзу включений модуль «Сертифікати», для створення й керування цифровими сертифікатами. Цифрові сертифікати

необхідні для того, щоб забезпечити можливість роботи мережних сервісів Інтернет шлюзу з використанням протоколу SSL. SSL (Secure Sockets Layer – рівень захищених сокетів) – криптографічний протокол, що забезпечує встановлення безпечного з'єднання між клієнтом і сервером. Використання протоколу SSL може бути продиктовано бажанням підвищити безпека роботи з даними й підвищенням рівня надійності й захищеності мережних сервісів.

Для додаткового контролю над безпекою Інтернет шлюзу, використовується модуль «Монітор файлів», що дозволяє відслідковувати зміни у важливих системних файлах і файлах конфігурації. Це може бути корисно в тому випадку, якщо зловмисник зміг яким-небудь образом одержати доступ до Інтернет шлюзу й зробив підміну файлів або ж змінив їх. У такому випадку в журналі буде запис про те, що є зміни в тих файлах, які відслідковуються.

Гарною підмогою в роботі системному адміністраторові послужить модуль «ARP-таблиця», що дозволяє відслідковувати появу в мережі нових пристроїв (за MAC-адресою), відповідність IP і MAC адрес, а також здійснювати прив'язку й відв'язку IP адрес до MAC адрес. Особливо корисний буде цей модуль у великій мережі. А також для відстеження несумлінних користувачів, які можуть змінювати MAC-адресу своїх мережних пристроїв, що створює додаткові проблеми IT-персоналу.

На додаток до цього, в Інтернет шлюзі є модуль «Мережні утиліти», у якому зібрані найбільше часто використовувані мережні утиліти для контролю стану мережі й діагностики мережних неполадок, а саме:

– Ping, для перевірки з'єднання.

– Traceroute, для відображення маршруту проходження запиту до потрібного хосту.

– Dig, утиліта для тестування роботи DNS, шляхом посилки різних запитів до них.

– Whois, для одержання інформації про власника домена або діапазону ip-адрес.

– Route, для відображення поточної таблиці маршрутизації Інтернет шлюзу.

– Ifconfig, показує поточне налаштування мережних інтерфейсів Інтернет шлюзу.

– Tcpdump, відображає заголовки пакетів, які проходять через задану мережну карту. Це дозволяє детально аналізувати проблеми з мережним устаткуванням і налаштуваннями Інтернет шлюзу.

– Також доступно ARP-сканування мережі (на доступність локальних пристроїв у мережі) і сканування на перевірку відкритих портів у мережі. Це дозволяє підвищити безпека мережі в цілому, шляхом виявлення проблем у налаштуваннях програмного забезпечення, що працює з мережею. Для забезпечення цих можливостей використовується широко відоме програмне забезпечення Nmap.

Модуль «Маршрути» дозволяє створювати гнучкі правила маршрутизації трафіку. В Інтернет шлюзі підтримується маршрутизація TCP/UDP, ICMP, GRE.

Функціональні модулі захисту від атак

Як і годиться комплексу подібного класу, для захисту від мережних атак ззовні використовується Інтернет шлюз і детектор виявлення атак.

Керування Інтернет шлюзом і робота із правилами дуже прості, завдяки продуманості веб-консолі. При установці Інтернет шлюзу, ряд правил додається автоматично, для коректного функціонування ряду мережних сервісів.

Крім звичайних розв'язних і заборонних правил, можна задавати правила маршрутизації, правила обмеження смуги пропущення й правила пріоритезації трафіку. Це дозволяє побудувати дуже гнучку систему обробки мережного трафіку. Керування пріоритетом трафіку дуже корисно у випадку високої завантаженості каналів або наявності в мережі критично важливих мережних сервісів, для яких важливо забезпечити постійну доступність. Приміром, якщо є веб-сервер, на якому розміщений сайт компанії й важливо, щоб клієнти завжди мали до нього доступ, то можна задати правило обробки трафіку, відповідно до

якого буде оброблятися трафік цього сервера в першу чергу. А от поштовому трафіку або FTP, можна задати й знижений пріоритет. До того ж, правилам можна задавати час дії, що саме по собі дуже зручно. І надає ще більшу гнучкість у плані керування мережним трафіком.

Для забезпечення додаткових можливостей керування мережним трафіком, Інтернет шлюз забезпечує можливість перенапрямку портів.

Для додаткового захисту від мережних атак використовується модуль «Детектор атак», реалізований за допомогою відомої IPS Snort, здатної виконувати реєстрацію пакетів і в режимі реального часу здійснювати аналіз трафіку в IP мережах. IPS Snort виконує протоколювання, аналіз, пошук по вмісту, а також широко використовується для активного блокування або пасивного виявлення цілого ряду нападів і зондувань, таких як переповнення буфера, стелс-сканування портів, атаки на веб-додатка, SMB-зондування й спроби визначення ОС.

Для коректної роботи цього модуля необхідно встановити правила для системи виявлення атак.

Для одержання й постійного відновлення першого списку правил, необхідно зареєструватися на сайті snort.org, одержати код Oinkcode для завантаження правил і ввести його в поле «Код для Oinkcode».

Другий список правил поширюється вільно без реєстрації.

Після того, як потрібні джерела списків правил відзначені в чек-боксах, можна нажати кнопку Обновити й чекати відновлення списків правил. Після чого система виявлення вторгнень готова до роботи.

Також можна вказати додаткові налаштування для цього модуля, список портів, що захищаються, мережні інтерфейси та інше

Варто відзначити, що Snort є досить потужною системою запобігання вторгнень і широко поширена.

Для перевірки трафіку на наявність шкідливого коду, в Інтернет шлюзі використовуються антивірусні продукти.

Використання антивірусних продуктів на шлюзі, дозволяє сильно підвищити безпека в локальній мережі. При цьому можливі кілька сценаріїв організації безпеки:

– Мережний трафік сканується на інтернет-шлюзі, на робочих місцях використовується антивірусне програмне забезпечення без модуля перевірки трафіку.

– Мережний трафік сканується на інтернет-шлюзі, на робочих місцях використовується антивірусне програмне забезпечення іншого виробника з модулем перевірки трафіку.

– Мережний трафік сканується на інтернет-шлюзі, на обраних робочих місцях використовується антивірусне програмне забезпечення іншого виробника з модулем перевірки трафіку.

Як бачимо, Інтернет шлюз має у своєму составі потужні засоби протидії мережним і вірусним атакам. Що дозволяє створити потужну інфраструктуру інформаційної безпеки організації або вільно інтегрувати його в існуючу й підвищити рівень безпеки в цілому.

Контроль доступу до ресурсів Інтернет

Тепер розглянемо, яким образом здійснюється контроль доступу користувачів до різних ресурсів мережі Інтернет.

У наш час, контроль доступу до ресурсів мережі Інтернет необхідний по різних причинах. Це може бути й бажання адміністрації підвищити продуктивність праці шляхом блокування розважальних ресурсів, соціальних мереж і іншого. Або підвищення рівня безпеки в локальній мережі шляхом блокування відомих шкідливих ресурсів. Або ж блокування доступу до небажаних ресурсів у дитячих освітніх установах згідно. Природно, база таки ресурсів повинна постійно поповнюватися.

Така можливість реалізована в Інтернет шлюзі. Варто відзначити, що ці список ресурсів у цих категоріях дуже великий.

Для більше гнучкого й повного контролю над доступом до ресурсів мережі Інтернет, в Інтернет шлюзі передбачена можливість створення власних категорій і автоматичних категорій.

Автоматичні категорії зручні тим, що системний адміністратор може сам звістки списки необхідних ресурсів, або ж брати готові, указувати Інтернет шлюзу місце розташування файлів зі списком ресурсів, а потім указати період автоматичного відновлення цієї категорії. Таким чином, якщо у файл зі списком будуть внесені нові дані, вони автоматично завантажаться в потрібну категорію на сервері.

Для того, щоб реалізувати контроль доступу до ресурсів мережі, застосовуються правила й набори правил для користувачів у мережі. Для роботи із правила служить модуль «Набори правил» розділу «Користувачі й статистика».

Є три передвстановлених набори правил (по-умовчанню самі правила там відсутні), а також можна створювати свої набори, для більшої наочності й спрощення роботи. Приміром, можна створити набір правил «Школа», де створити правила заборонні відвідувати ті або інші категорії ресурсів небажаних для дітей.

Доступ до ресурсів із зазначених категорій можна заборонити або, при спробі відвідати ресурс із забороненої категорії, перенаправляти на потрібний сайт. Приміром, на сайт школи.

Правила можуть мати певний час дії, яке можна задати. Це дозволяє гнучко управляти контролем доступу до ресурсів залежно від часу доби. Приміром, можна заборонити відвідування соціальних мереж у робочий час, але дозволяти відвідувати в обідню перерву.

Крім правил контролю доступу, можна створювати правила керування шириною пропущення каналу для окремих користувачів, груп користувачів і діапазонів адрес. Також, в Інтернет шлюзі передбачена можливість перегляду детальної статистики відвідування ресурсів мережі, обсяг спожитого трафіку й іншої статистичної інформації.

Таким чином, Інтернет шлюз, надає можливість контролювати роботу користувачів у мережі Інтернет у повному обсязі. З урахуванням контролю доступу до ресурсів мережі, обмеження на обсяг споживаного трафіку й керування завантаженням каналів.

Моніторинг і керування системою

Крім всіх перерахованих вище можливостей, Інтернет шлюз дозволяє в реальному часі контролювати стан устаткування, на якому він установлений, переглядати завантаження каналів, якість зв'язку, контролювати стан операційної системи й багато чого іншого. Для контролю всіх цих параметрів служить модуль «Моніторинг»

Дані сортуються по часових інтервалах: остання година, останніх 6 годин, останній день, тиждень і місяць. Це дає системному адміністраторові можливість стежити за станом системи в цілому, її окремих параметрів, відслідковувати пікове навантаження для можливої оптимізації роботи системи й запобігання відмови в обслуговуванні.

Для полегшення відновлення системи після збою (від подібного не застрахований ніхто), в Інтернет шлюзі передбачена можливість автоматичного створення резервних копій налаштувань системи, а також повного резервного копіювання системи. Крім локального зберігання резервних копій, передбачена можливість збереження резервних копій на флеш-накопичувач і на вилучений FTP-сервер. Що сильно підвищує можливість швидкого відновлення роботи, у випадку якщо локальні резервні копії були загублені або ушкоджені. Налаштування резервного копіювання можна провести в модулі «Резервне копіювання».

Для додаткового контролю Інтернет шлюзу передбачений модуль «Системний журнал». У модулі «системний журнал» відображаються повідомлення про дії користувачів, змінах у статусах сервісів і помилках системи.

Для того, щоб не пропустити яке-небудь дуже важливу подію, у цьому модулі передбачена система відправлення повідомлень відповідальній особі.

Повідомлення може бути відправлене по електронній пошті, а також за допомогою Jabber або ICQ.

Таким чином, немає необхідності постійно стежити за журналом подій, що, у цілому, важко. Така можливість забезпечує своєчасне повідомлення про ті або інші події, без постійного відволікання уваги.

Модуль «Жорсткі диски» призначений для керування дисковою підсистемою сервера із установленим Інтернет шлюзом. Досить часто виходить так, що обсягу встановлених жорстких дисків перестає вистачати. Для цього й була уведена можливість легкого додавання нових жорстких дисків, створення додаткових розділів і іншого в Інтернет шлюзі ІКС.

Для тих випадків, коли доступ до Інтернет шлюзу неможливий за допомогою веб-інтерфейсу, передбачена консоль відновлення, яку можна активувати натисканням клавіші F2, на клавіатурі, що приєднана до сервера із установленим Інтернет шлюзом.

За допомогою консолі відновлення можна змінити налаштування мережних інтерфейсів, змінити таблицю маршрутизації, змінити паролі доступу до Інтернет шлюзу, настроїти дискову підсистему, перезавантажити або виключити сервер.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема системи. На який розглянута розроблена система в цілому.

Система контролю Інтернет шлюзів на базі ОС Ubuntu – цілий комплекс рішень, проксі (proxy) сервер, що дозволяє:

– захищати мережу організації від зовнішніх атак;

– роздавати інтернет користувачам мережі;

– використовувати аварійне резервування каналів інтернет;

– повністю контролювати трафік по всіх портах і протоколам;

– використання двох провайдерів і більше;

– централізоване керування філіями організації;

– можливість розгортання у віртуальних середовищах;

– обмежувати доступ користувачів до ресурсів розважального характеру;

– блокування сайтів по списках;

– балансувати трафік між співробітниками для більше якісного й ощадливого використання каналу;

– обмеження швидкості користувачам;

– блокувати вірусну активність;

– безкоштовна консультація по першому старті системи;

– безкоштовна версія для 6 користувачів мережі, малого бізнесу;

– поштовий сервер, NAT, високоякісний шейпер і багато чого іншого**.**

Система контролю Інтернет шлюзів на базі ОС Ubuntu у частині реалізації біллингу реалізується для:

– для інтернет провайдерів, готелів, орендодавців, інтернет-кафе;

– для надання клієнтам швидке підключення по картах оплати;

– переведення коштів через термінали оплати;

– високоякісного шейпера обмежуючу смугу пропущення на безлімітних і лімітних каналах;

– контроль трафіку;

– облік трафіку;

– використання декількох вхідних каналів від вищестоящого провайдеру;

– використання двох провайдерів і більше;

– можливість розгортання у віртуальних середовищах;

– централізованого керування серверами доступу навіть у різних містах;

– поштовий сервер, NAT, антиспам і безліч інших функцій;

– блокування сайтів по списках;

– безкоштовна консультація по першому старті системи;

– відкритий API системи**.**

Облік трафіку й балансування:

– Шейпер високої точності за власною технологією upit-systems.

– Твердий контроль по всіх портах і протоколах.

– Миттєве блокування по закінченню позитивного балансу (ні однієї копійки не буде витрачено понад ліміт).

– Гнучкі тарифні й офіс плани.

– Монітори реального часу.

– Робота з термінальними серверами.

Маршрутизатор:

– Маршрутизацію по декількох каналах вхідного трафіку.

– Активний контроль стану каналів інтернет, щопоставляються від вищестоящих провайдерів.

– Аварійний перехід на резервний (резервні) канали при відсутності зв'язку по одному з каналів.

Файрвол (міжмережний екран):

– Блокування по портах від небажаного трафіку.

– Систему антифлуд (блокування вірусної активності).

– Захист від сканування портів.

– Захист сервера від зовнішніх атак.

– Захист Вашої локальної мережі від зовнішніх погроз.

Контроль дзвінків Міні-АТС:

– Систему контролю дзвінків Міні-АТС.

– Звітність як на зовнішні дзвінки так і на внутрішні.

– Монітори реального часу.

Низькі вимоги до ресурсів: сервер можна використовувати для безлічі додатків таких як: файлове сховище, поштовий сервер, сервер 1С, системи резервного копіювання й багатьох інших.

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. На який детально розглянута розроблена система. Розглянемо роботу розробленої системи на прикладі запиту Інтернету користувача № N.

Користувач N через локальну мережу через WEB браузер робочої станції посилає запит до мережі з даними IP адреси. На сервері розроблене ПЗ проводить аутентифікацію робочої станції за допомогою протоколу тунелюваня та далі проводе запит сторінки через брандмауер сервера. В цей час розроблене ПЗ проводе зчитування трафіку локальної мережі, через демон PPTP, керування

відбувається через відповідний модуль. Всі дані роботи користувача у Інтернеті заносяться до бази даних сервера. Основні таблиці це «Таблиця Кошти» де зберігаються дані витрачених коштів, «Таблиця Користувачі» де знаходиться інформація користувача, «Таблиця Статистика» статистичні дані та «Таблиця Група» де зберігаються встановлені привілегії користувача.

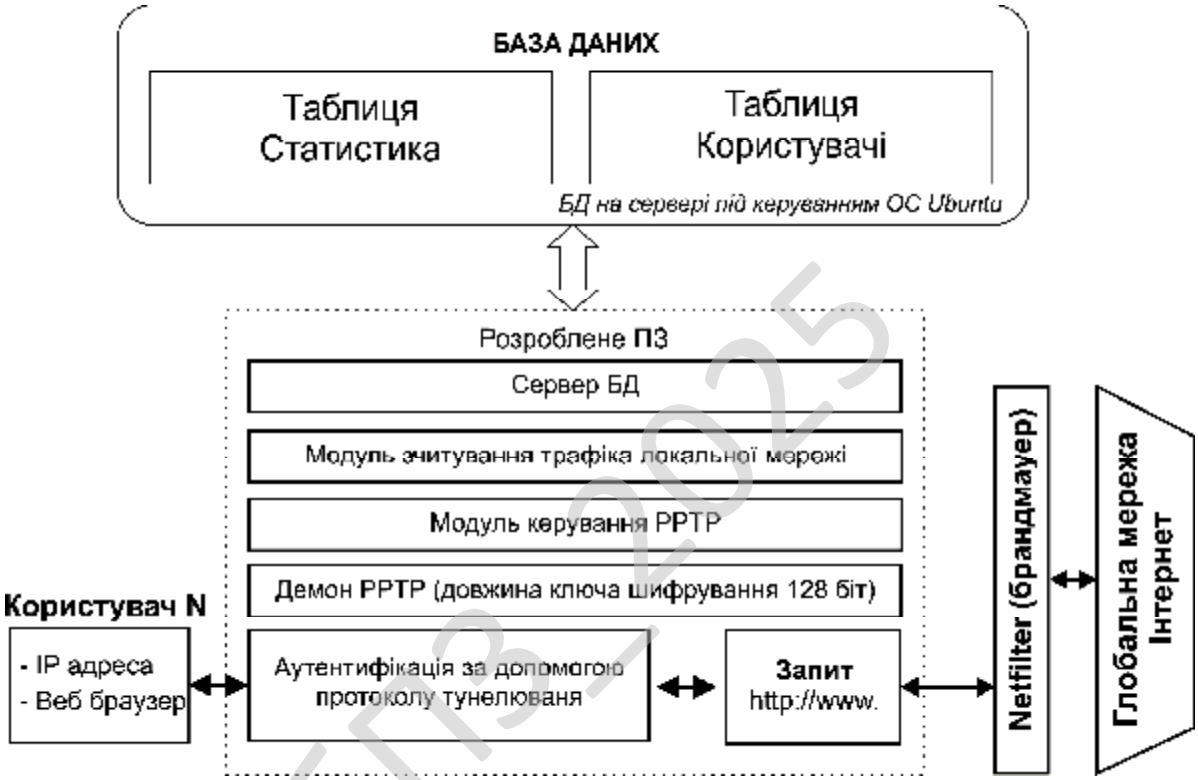

Рисунок 3.2 – Функціональна схема системи

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

3.4 Розробка діаграми процесів

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

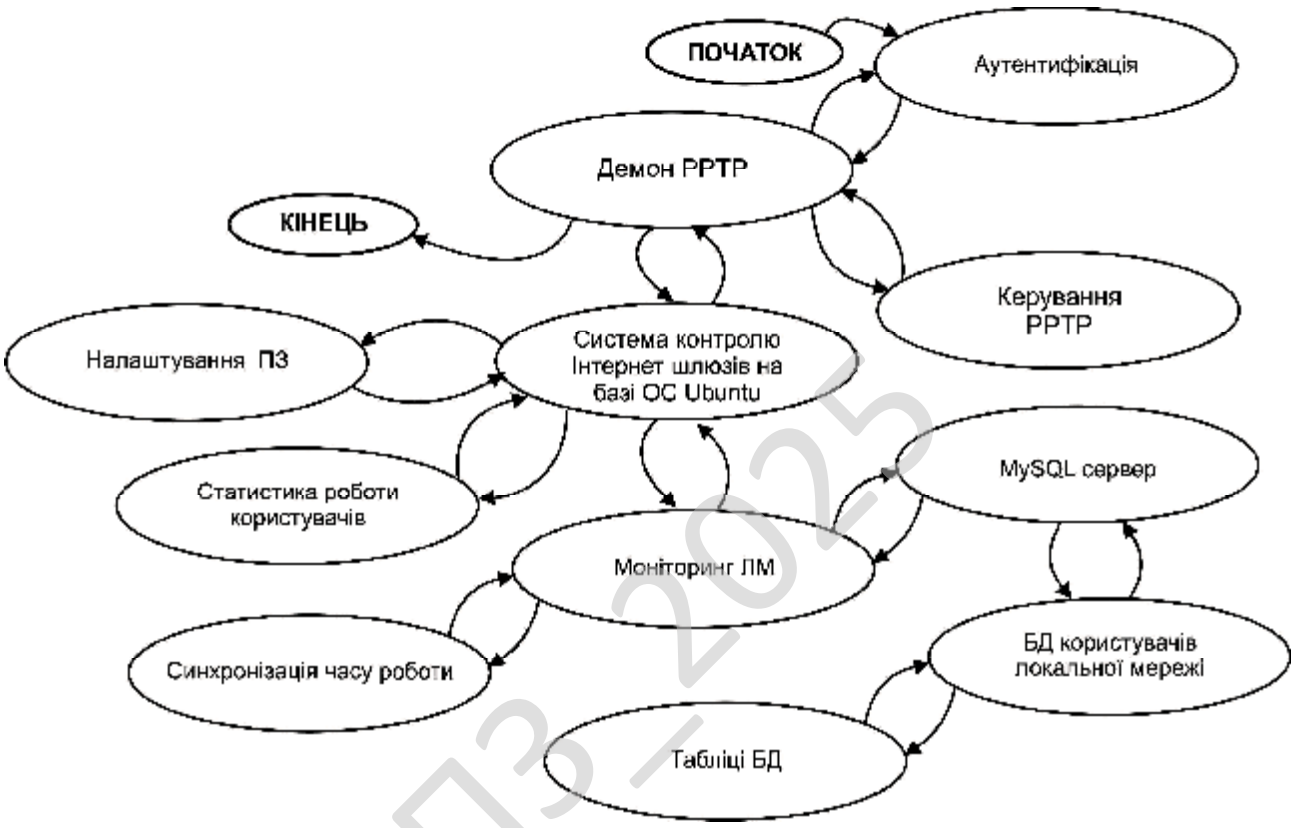

Рисунок 3.3 – Діаграма взаємодії процесів

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

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

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

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

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

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

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

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

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

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

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.

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.

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.

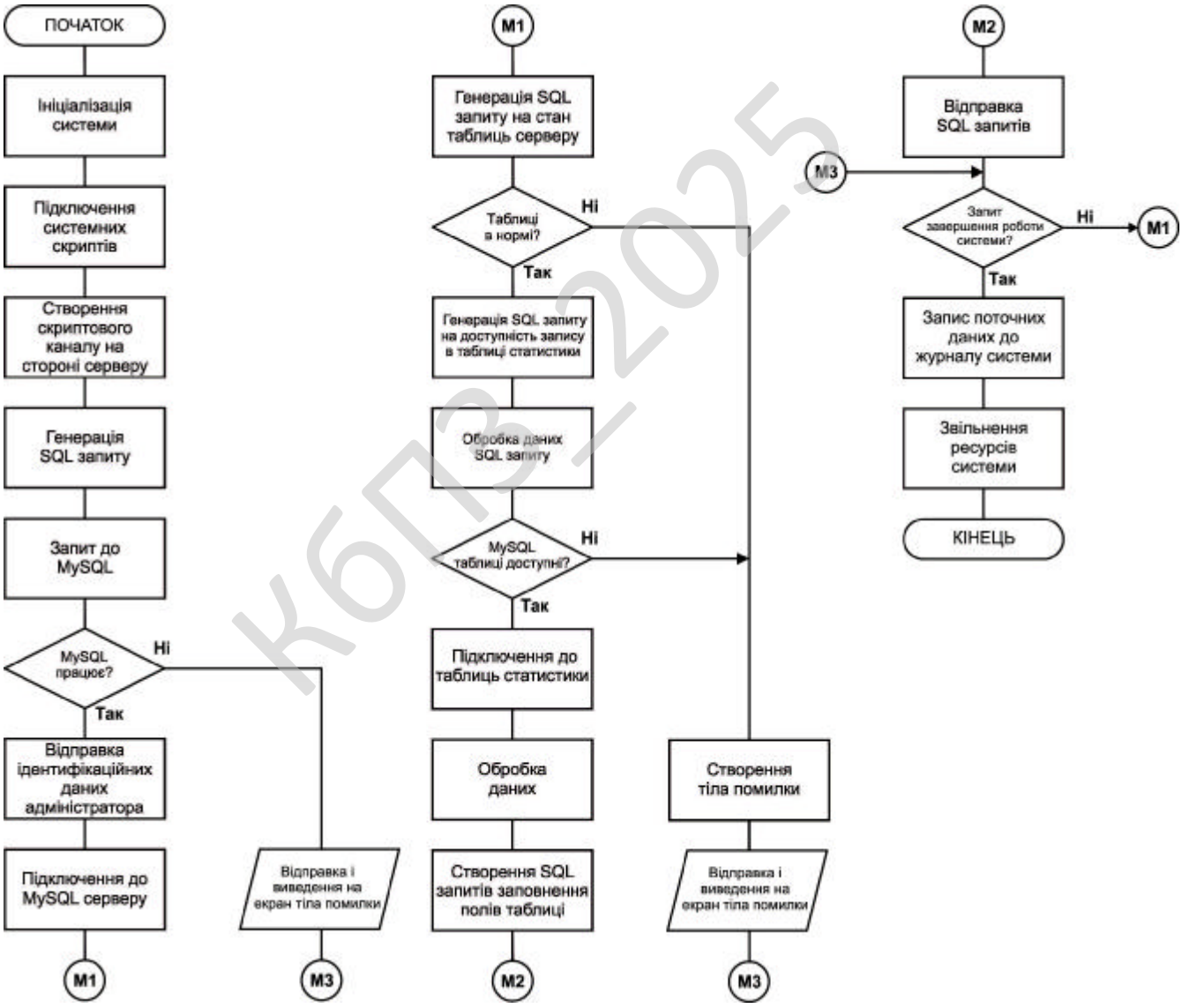

Рисунок 4.1 – Блок-схема основної програми

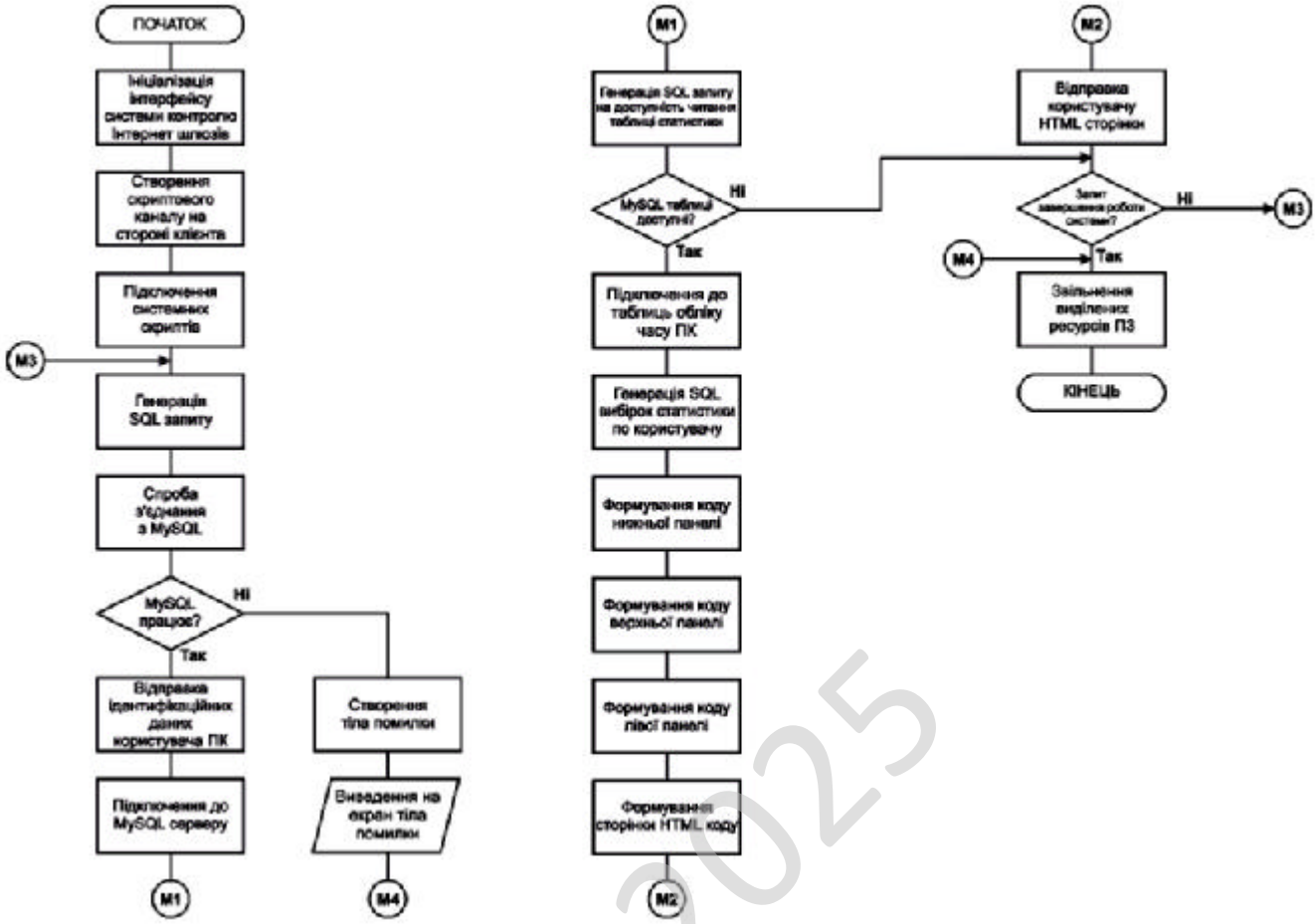

Рисунок 4.2 – Блок-схема роботи підпрограми

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 контролю Інтернет шлюзів на базі ОС Ubuntu.

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.

Основні технічні моменти

Система контролю інтернет шлюзів на базі операційної системи Ubuntu забезпечує моніторинг, фільтрацію та керування мережею для локальної інфраструктури.

Система розробляється у рамках випускної кваліфікаційної роботи та реалізується з використанням мови програмування Python. Вона функціонує як серверне рішення, яке інтегрується безпосередньо в інфраструктуру маршрутизації або працює паралельно з основним маршрутизатором.

Основною метою є виявлення критичних відхилень у мережевому трафіку, обмеження небажаного доступу, фіксація підозрілих з'єднань та ведення журналу всіх з'єднань з подальшим аналізом.

Система отримує доступ до мережевого інтерфейсу через інструменти низького рівня, такі як psutil, socket, subprocess та scapy, та використовує логічну обробку даних у реальному часі.

Візуалізація результатів виконується за допомогою бібліотек matplotlib та tkinter.

Архітектура ПЗ

Архітектура програми складається з декількох взаємопов’язаних модулів.

Основний модуль відповідає за моніторинг інтерфейсів, зчитування ARP-таблиць, визначення активних підключень за допомогою netstat, фільтрацію небезпечних IP-адрес та створення логів.

Модуль взаємодії з користувачем забезпечує консольний або графічний інтерфейс, який дозволяє виводити список поточних підключень, стан інтерфейсів та статистику передачі.

Модуль контролю використовує дані для аналізу перевантаження шлюзу, оцінки затримок та виявлення атак типу DoS або ARP spoofing.

Основна функція системи main() викликає ініціалізацію моніторингового циклу, запускає процес перевірки мережевого навантаження та контролює роботу підсистем логування.

Компонент get_active_connections() зчитує інформацію про з'єднання з /proc/net/tcp та /proc/net/udp.

Модуль log_traffic() створює текстовий файл із часовими мітками та інформацією про відправника, отримувача та обсяг переданих даних.

Функція analyze_traffic() оцінює частоту з'єднань і виділяє пікові значення. При перевищенні порогу відбувається або автоматичне блокування IP-адреси через iptables, або виведення повідомлення адміністратору системи.

Для глибокої перевірки трафіку застосовується бібліотека scapy, яка дозволяє фільтрувати пакети за MAC-адресами та протоколами.

У разі виявлення аномалій система автоматично створює звіт у вигляді CSV файлу та відображає попередження у терміналі.

У структурі системи застосовується модульний принцип, що дає змогу змінювати компоненти без впливу на загальну функціональність.

Наприклад, модуль report_generator можна підключати окремо у вигляді зовнішнього сценарію Python для формування щотижневих PDF-звітів за допомогою бібліотеки reportlab.

Для забезпечення коректності обраних рішень було проведено експериментальні випробування у локальній мережі на п'яти вузлах.

Швидкість реакції на зростання кількості з'єднань оцінюється в середньому як 120 мілісекунд.

Навантаження на процесор не перевищує 7 відсотків під час фонової роботи.

Пропускна здатність мережевого інтерфейсу з активним моніторингом залишається в межах 96 відсотків від номінального значення, що свідчить про ефективність оптимізації коду.

Розрахунки виконуються для оцінки обсягу логів, обчислювальних витрат та часу реагування.

Наприклад, при середній частоті створення 100 з'єднань за хвилину, обсяг логів за добу становить близько 2 мегабайт.

Це дозволяє системі працювати автономно при наявності мінімум 200 мегабайт доступного простору для логів на тиждень.

Згідно з тестуванням на Ubuntu 22.04, система працює стабільно понад 72 години без потреби у перезапуску процесів.

Висновок, що випливає з даних, підтверджує обґрунтованість використання Python для реалізації контрольної системи шлюзів. Його бібліотеки дозволяють ефективно керувати мережевими інтерфейсами, зберігати журнали та реалізовувати динамічну реакцію на потенційні загрози.

Вибір операційної системи Ubuntu забезпечує стабільність, доступність інструментів для адміністрування та сумісність з інструментами безпеки.

Розглянемо частину вихідного коду, який реалізує основні функції системи контролю інтернет шлюзів на Ubuntu. Він включає моніторинг активних підключень, фіксацію трафіку, виявлення аномалій та логування.

```
import os
import time
import socket
import psutil
import subprocess
from datetime import datetime

LOG_FILE = "/var/log/gateway_monitor.log"
TRAFFIC_THRESHOLD = 100

def get_active_connections():
```

```
    connections = psutil.net_connections(kind='inet')
    active = []
    for conn in connections:
        if conn.status == 'ESTABLISHED':
            laddr = f"{conn.laddr.ip}:{conn.laddr.port}"
            raddr = f"{conn.raddr.ip}:
                    {conn.raddr.port}" if conn.raddr else "N/A"
            active.append((laddr, raddr))
    return active

def log_traffic(connections):
    with open(LOG_FILE, 'a') as log:
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        for local, remote in connections:
            log.write(f"{timestamp} {local} -> {remote}\n")

def analyze_traffic(connections):
    ip_count = {}
    for _, remote in connections:
        ip = remote.split(':')[0]
        if ip != 'N/A':
            ip_count[ip] = ip_count.get(ip, 0) + 1

    suspicious = [ip for ip, count in ip_count.items() if
                  count > TRAFFIC_THRESHOLD]
    return suspicious

def block_ip(ip):
    command = f"sudo iptables -A INPUT -s {ip} -j DROP"
    subprocess.call(command, shell=True)

def monitor_loop():
    while True:
        connections = get_active_connections()
        log_traffic(connections)
        suspicious_ips = analyze_traffic(connections)
        for ip in suspicious_ips:
            block_ip(ip)
        time.sleep(60)

def main():
    if os.geteuid() != 0:
```

```
            print("Скрипт необхідно запускати з правами адміністратора")
            return
        print("Запуск моніторингу шлюзу...")
        monitor_loop()
main()
```

1. Ініціалізація моніторингу. Запускається основна функція main(), яка перевіряє права адміністратора та ініціює безперервний цикл monitor_loop.

2. Збір активних підключень. Функція get_active_connections() отримує всі з'єднання типу 'inet' у стані 'ESTABLISHED' через бібліотеку psutil.

3. Логування трафіку. Функція log_traffic() записує в лог-файл усі активні з'єднання разом із міткою часу.

4. Виявлення підозрілих IP. Функція analyze_traffic() підраховує частоту з'єднань з кожним IP і повертає список тих, що перевищують встановлений поріг.

5. Блокування IP. Функція block_ip() викликає iptables, щоб заборонити вхідний трафік з підозрілих адрес.

Цей код формує ядро системи та може бути розширений за рахунок аналізу ARP-таблиці, перевірки MAC-адрес, інтеграції графічного інтерфейсу або віддаленого керування шлюзом.

4.2 Захист розробленого програмного забезпечення

Tiny Encryption Algorithm (TEA) [1] – блочний алгоритм шифрування типу «Мережі Фейстеля». Алгоритм був розроблений на факультеті комп'ютерних наук Кембриджського університету Девідом Вілером (David Wheeler) і Роджером Нідгемом (Roger Needham) та вперше представлений в 1994 році [2] на симпозіумі зі швидкими алгоритмами шифрування в Льовені (Бельгія).

Шифр не патентований, широко використовується в ряді криптографічних додатків і широкому спектрі апаратного забезпечення, завдяки вкрай низькими вимогами до пам'яті й простоті реалізації.

Алгоритм має як програмну реалізацію на різних мовах програмування, так і апаратну реалізацію на інтегральних схемах типу FPGA.

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму TEA, який заснований на бітових операцій з 64-бітним блоком, має 128-бітний ключ шифрування. Стандартна кількість раундів мережі Фейстеля біля 64 (32 циклу), однак, для досягнення найкращої продуктивності або шифрування, число циклів можна варіювати від 8 (16 раундів) до 64 (128 раундів). Мережа Фейстеля несиметрична через використання в якості операції накладення додавання за модулем 2^{32}.

Перевагами шифру є його простота в реалізації, невеликий розмір коду й досить висока швидкість виконання, а також можливість оптимізації виконання на стандартних 32-бітних процесорах, так як в якості основних операцій використовуються операції виключна «АБО» (XOR), побітового зсуву й додавання за модулем 2^{32}. Оскільки алгоритм не використовує таблиць підстановки і раундова функція досить проста, алгоритму потрібно не менше 16 циклів (32 раундів) для досягнення ефективної дифузії, хоча повна дифузія досягається вже через 6 циклів (12 раундів).

Алгоритм має відмінну стійкість до лінійного криптоаналізу і досить гарну до диференціального криптоаналізу. Головним недоліком цього алгоритму шифрування є його вразливість до атак «на пов'язаних ключах» (англ. Related-key attack). Через простий розклад ключів кожен ключ має 3 еквівалентних ключа. Це означає, що ефективна довжина ключа складає всього 126 біт [3] [4], тому даний алгоритм не слід використовувати в якості геш-функції.

Опис алгоритму

Вихідний текст розбивається на блоки по 64 біта кожен. 128-бітний ключ K ділиться на чотири 32-бітних підключа K [0], K [1], K [2] і K [3]. На цьому підготовчий процес закінчується, після чого кожен 64-бітний блок шифрується протягом 32 циклів (64 раундів) за нижченаведеним алгоритмом. [5]

Припустимо, що на вхід n-го раунду ($1 \leq n \leq 64$) надходять права й ліва частини (L_n, R_n), тоді на виході n-го раунду будуть ліва й права частини (L_{n+1}, R_{n+1}), які обчислюються за такими правилами:

$$L_{n+1} = R_n.$$

Якщо n = 2 * i – 1 для 1 ≤ i ≤ 32 (непарні раунди), то:

$$R_{n+1} = L_n (\{ [R_n 4] K [0] \} \{ R_n i * \delta \} \{ [R_n 5] K [1] \}).$$

Якщо n = 2 * i для 1 ≤ i ≤ 32 (парні раунди), то:

$$R_{n+1} = L_n (\{ [R_n 4] K [2] \} \{ R_n i * \delta \} \{ [R_n 5] K [3] \}).$$

Де

X Y – операція додавання чисел X і Y за модулем 232.

X Y – побітове виключне АБО» (XOR) чисел X і Y, яке в мові програмування Сі позначається як X ^ Y

X Y і X Y – операції побітового зсуву числа X на Y біт вліво й вправо відповідно.

Константа δ була виведена з Золотого перерізу:

$$\delta = (– 1) * 2^{31} = 2654435769 = 9E3779B9_h.$$

У кожному раунді константа множиться на номер циклу i. Це було зроблено для запобігання простих атак, заснованих на симетрії раундів.

Також очевидно, що в алгоритмі шифрування TEA немає як такого алгоритму розкладу ключів. Замість цього в непарних раундах використовуються підключі K [0] та [1], у парних – K [2] і [3].

Так як це блочний шифроалгоритм, де довжина блоку 64-біт, а довжина даних може бути не кратна 64-біт, значення всіх байтів, які доповнюють блок до кратності в 64-біт, встановлюється в 0x01.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене ПЗ немає графічного інтерфейсу, поточні результати її роботи зберігаються у файлі system-data.tt як це можна побачити на рисунку 5.1. Під час роботи розроблене ПЗ системи контролю Інтернет шлюзів на базі ОС Ubuntu проводить контроль наступних підсистем:

– Контроль доступу до локального WEB-серверу.

– Контроль доступу до локального FTP-серверу.

– Декілька каналів вхідного трафіку.

– Міжмережний екран.

– Локальний сервер електронної пошти.

– Контроль доступу до Інтернету.

– Контроль телефонних дзвінків.

– Модуль біллінгу.

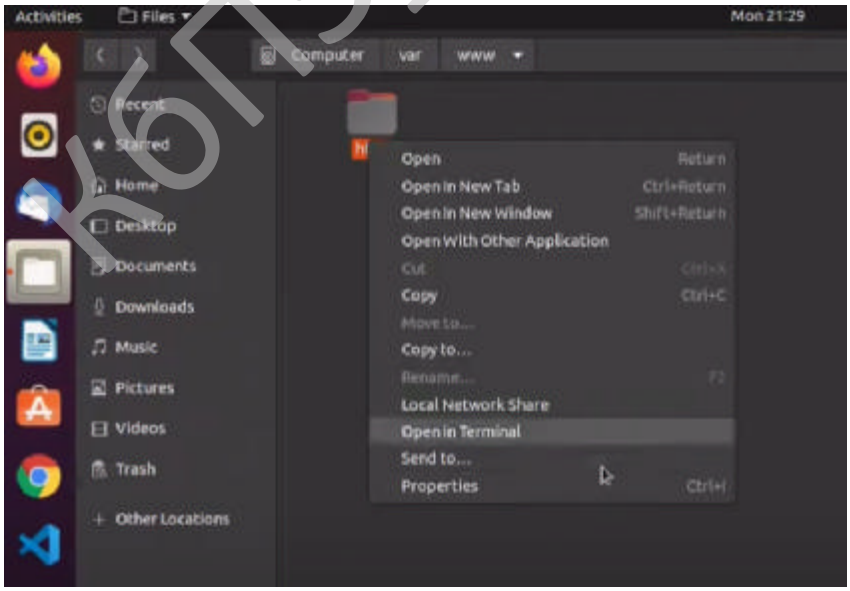

Рисунок 5.1 – Результат роботи ПЗ

Налаштування розробленої системи контролю Інтернет шлюзів на базі ОС Ubuntu знаходиться у файлі settings.conf.

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 ITc).

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

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

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

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 – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи контролю Інтернет шлюзів на базі ОС Ubuntu.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем контролю Інтернет шлюзів на базі ОС Ubuntu.

– Досліджена система контролю Інтернет шлюзів на базі ОС Ubuntu.

– На основі отриманих результатів досліджень створена програмна реалізація системи контролю Інтернет шлюзів на базі ОС Ubuntu.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання контролю Інтернет шлюзів на базі ОС Ubuntu.

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

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

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи контролю Інтернет шлюзів на базі ОС Ubuntu. 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 Ubuntu.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм TEA.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

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

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

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

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

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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).

43. 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

44. 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.

45. 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.

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

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

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

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

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

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

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

53. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи контролю Інтернет шлюзів на базі ОС Ubuntu.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 47-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи контролю Інтернет шлюзів на базі ОС Ubuntu.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи контролю Інтернет шлюзів на базі ОС Ubuntu;

– 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 Ubuntu і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 63 аркуші.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 5.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Марченко К.М.

Програмне забезпечення системи контролю Інтернет шлюзів на базі ОС Ubuntu

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 21

Літера: РП

Кропивницький – 2025 року

Основна програма

```
import os
import sys
import time
import subprocess
import socket
import psutil
import datetime
import logging
import threading
import json
import shutil

# Ініціалізація логування
logging.basicConfig(
    filename='gateway_monitor.log',
    level=logging.DEBUG,
    format='%(asctime)s %(levelname)s:%(message)s'
)

# Функція для перевірки наявності інтернет-з'єднання
def check_internet_connection():
    try:
        # Спроба підключитися до DNS Google
        socket.setdefaulttimeout(3)
        host = socket.gethostbyname("8.8.8.8")
        s = socket.create_connection((host, 53), 2)
        s.close()
        return True
    except Exception as e:
        logging.error(f"Connection failed: {str(e)}")
        return False

# Функція для отримання IP адреси шлюзу
def get_default_gateway():
    try:
        # Отримання шлюзу через ip route
        result = subprocess.check_output("ip route show default", shell=True)
        result = result.decode('utf-8')
        lines = result.split('\n')
        for line in lines:
            if "default via" in line:
                parts = line.split()
                gateway = parts[2]
                return gateway
        return None
    except Exception as e:
        logging.error(f"Get default gateway error: {str(e)}")
        return None

# Функція для перевірки стану шлюзу
def ping_gateway(gateway_ip):
    try:
        # Виконання ping до шлюзу
        response = subprocess.call(['ping', '-c', '3', gateway_ip],
                                   stdout=subprocess.DEVNULL,
                                   stderr=subprocess.DEVNULL)
        if response == 0:
            return True
        else:
            return False
    except Exception as e:
```

```
        logging.error(f"Ping gateway error: {str(e)}")
        return False

# Функція для збереження логів з мережею
def log_network_status():
    gateway = get_default_gateway()
    if gateway:
        online = ping_gateway(gateway)
        if online:
            logging.info(f"Gateway {gateway} is reachable")
        else:
            logging.warning(f"Gateway {gateway} is not reachable")
    else:
        logging.error("No default gateway found")

# Функція для моніторингу мережевих інтерфейсів
def monitor_network_interfaces():
    interfaces = psutil.net_if_addrs()
    stats = psutil.net_if_stats()
    for interface in interfaces:
        is_up = stats[interface].isup if interface in stats else False
        logging.info(f"Interface: {interface}, Status: {'UP' if is_up else
'DOWN'}")

# Функція для отримання статистики мережевого трафіку
def get_network_io():
    io = psutil.net_io_counters(pernic=True)
    for iface, data in io.items():
        logging.info(f"Interface: {iface}, Sent: {data.bytes_sent} bytes,
Received: {data.bytes_recv} bytes")

# Функція для збору системної інформації
def get_system_info():
    uname = os.uname()
    logging.info(f"System: {uname.sysname}")
    logging.info(f"Node Name: {uname.nodename}")
    logging.info(f"Release: {uname.release}")
    logging.info(f"Version: {uname.version}")
    logging.info(f"Machine: {uname.machine}")

# Функція для перевірки конфігураційних файлів шлюзу
def check_gateway_config():
    config_paths = [
        "/etc/network/interfaces",
        "/etc/netplan/01-netcfg.yaml",
        "/etc/resolv.conf"
    ]
    for path in config_paths:
        if os.path.exists(path):
            with open(path, 'r') as file:
                content = file.read()
                logging.info(f"Contents of {path}:\n{content}")
        else:
            logging.warning(f"Configuration file {path} not found")

# Функція для збереження знімку статусу
def save_snapshot():
    snapshot = {}
    snapshot['timestamp'] = str(datetime.datetime.now())
    snapshot['gateway'] = get_default_gateway()
    snapshot['interfaces'] = psutil.net_if_stats()
    snapshot['connections'] = len(psutil.net_connections())
```

```
    snapshot['network_io'] = {iface: {'sent': data.bytes_sent, 'recv':
data.bytes_recv}
                              for iface, data in
psutil.net_io_counters(pernic=True).items()}
    with open('snapshot.json', 'w') as f:
        json.dump(snapshot, f, indent=4)

# Функція для виконання періодичного моніторингу
def start_periodic_monitoring():
    while True:
        check_internet_connection()
        log_network_status()
        monitor_network_interfaces()
        get_network_io()
        save_snapshot()
        time.sleep(60)

# Функція для виконання резервного копіювання логів
def backup_logs():
    try:
        timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
        backup_file = f"gateway_monitor_backup_{timestamp}.log"
        shutil.copy('gateway_monitor.log', backup_file)
        logging.info(f"Backup created: {backup_file}")
    except Exception as e:
        logging.error(f"Backup failed: {str(e)}")

# Основна точка входу
def main():
    logging.info("Starting Internet Gateway Control System")
    get_system_info()
    check_gateway_config()
    thread = threading.Thread(target=start_periodic_monitoring)
    thread.start()

# Запуск головної функції
if __name__ == '__main__':
    main()
```

Файл getdefaultgateway.py

```
import os
import sys
import time
import threading
import socket
import subprocess
import json
import psutil
import paramiko
from flask import Flask, request, jsonify, send_file
from telegram.ext import Updater, CommandHandler
import matplotlib.pyplot as plt
import io

TELEGRAM_TOKEN = os.getenv('TELEGRAM_TOKEN')
CHAT_ID = int(os.getenv('TELEGRAM_CHAT_ID', '0'))

app = Flask(__name__)

def get_default_gateway():
    result = subprocess.check_output("ip route show default",
shell=True).decode('utf-8')
    for line in result.split('\n'):
        if "default via" in line:
            return line.split()[2]
    return None

def ping_gateway(ip):
    return subprocess.call(['ping', '-c', '2', ip],
                           stdout=subprocess.DEVNULL,
                           stderr=subprocess.DEVNULL) == 0

def scan_devices(ip_range):
    from scapy.all import ARP, Ether, srp
    arp = ARP(pdst=ip_range)
    ether = Ether(dst="ff:ff:ff:ff:ff:ff")
    packet = ether/arp
    result = srp(packet, timeout=2, verbose=False)[0]
    devices = []
    for _, received in result:
        devices.append({'ip': received.psrc, 'mac': received.hwsrc})
    return devices

def ssh_command(host, port, user, pwd, cmd):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    client.connect(hostname=host, port=port, username=user, password=pwd)
    stdin, stdout, stderr = client.exec_command(cmd)
    output = stdout.read().decode()
    client.close()
    return output

@app.route('/')
def index():
    stats = psutil.net_if_stats()
    io_counters = psutil.net_io_counters(pernic=True)
    return jsonify({
        'interfaces': {iface: stats[iface].isup for iface in stats},
        'io': {iface: {'sent': io_counters[iface].bytes_sent, 'recv':
io_counters[iface].bytes_recv} for iface in io_counters}
    })
```

```
@app.route('/devices')
def devices():
    gateway = get_default_gateway()
    devices = scan_devices(f'{gateway}/24') if gateway else []
    return jsonify(devices)

@app.route('/traffic.png')
def traffic():
    plt.figure()
    io_counters = psutil.net_io_counters()
    labels = ['sent', 'recv']
    values = [io_counters.bytes_sent, io_counters.bytes_recv]
    plt.bar(labels, values)
    buf = io.BytesIO()
    plt.savefig(buf, format='png')
    buf.seek(0)
    return send_file(buf, mimetype='image/png')

@app.route('/execute')
def execute():
    host = request.args.get('host')
    user = request.args.get('user')
    pwd = request.args.get('pwd')
    cmd = request.args.get('cmd')
    output = ssh_command(host, 22, user, pwd, cmd)
    return jsonify({'output': output})

def start_bot():
    updater = Updater(TELEGRAM_TOKEN, use_context=True)
    def start(update, context):
        context.bot.send_message(chat_id=update.effective_chat.id, text='Gateway
Monitor Bot Started')
    updater.dispatcher.add_handler(CommandHandler('start', start))
    def check_gateway(context):
        gateway = get_default_gateway()
        if gateway and not ping_gateway(gateway):
            context.bot.send_message(chat_id=CHAT_ID, text=f'Gateway {gateway}
is down')
    job_queue = updater.job_queue
    job_queue.run_repeating(check_gateway, interval=60, first=0)
    updater.start_polling()
    updater.idle()

if __name__ == '__main__':
    threading.Thread(target=start_bot).start()
    app.run(host='0.0.0.0', port=5000)
```

Файл ConfigLoader.py

```
import yaml
import time
import threading
import subprocess
import json
from scapy.all import sniff, DNS, ARP, Ether, srp
import requests
from ftplib import FTP
import networkx as nx
import matplotlib.pyplot as plt
import csv

class ConfigLoader:
    def __init__(self, file_path='config.yaml'):
        self.file_path = file_path
        self.config = {}
        self.load_config()
    def load_config(self):
        with open(self.file_path) as f:
            self.config = yaml.safe_load(f)
    def get(self, key, default=None):
        return self.config.get(key, default)

class GatewayHistoryLogger:
    def __init__(self, gateways, log_file='gateway_history.log', interval=60):
        self.gateways = gateways
        self.log_file = log_file
        self.interval = interval
    def ping(self, address):
        return subprocess.call(['ping','-c','2',address],
stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL) == 0
    def log_status(self):
        timestamp = time.strftime('%Y-%m-%d %H:%M:%S')
        for gw in self.gateways:
            res = self.ping(gw)
            status = 'UP' if res else 'DOWN'
            entry = f'{timestamp},{gw},{status}'
            with open(self.log_file,'a') as f:
                f.write(entry+'\n')
    def run(self):
        while True:
            self.log_status()
            time.sleep(self.interval)

class DataExporter:
    def __init__(self, history_file='gateway_history.log',
export_json='export.json', export_csv='export.csv', interval=600):
        self.history_file = history_file
        self.export_json = export_json
        self.export_csv = export_csv
        self.interval = interval
    def export(self):
        entries = []
        with open(self.history_file) as f:
            for line in f:
                parts = line.strip().split(',')
                if len(parts) == 3:
                    entries.append({'time': parts[0], 'gateway': parts[1],
'status': parts[2]})
        with open(self.export_json, 'w') as f:
            json.dump(entries, f, indent=2)
```

```
        if entries:
            keys = entries[0].keys()
            with open(self.export_csv, 'w', newline='') as f:
                writer = csv.DictWriter(f, fieldnames=list(keys))
                writer.writeheader()
                writer.writerows(entries)
    def run(self):
        while True:
            self.export()
            time.sleep(self.interval)

class DNSQueryAnalyzer:
    def __init__(self, output_file='dns_queries.json'):
        self.output_file = output_file
    def callback(self, packet):
        if packet.haslayer(DNS) and packet.getlayer(DNS).qr == 0:
            q = packet.getlayer(DNS).qd.qname.decode()
            t = time.strftime('%Y-%m-%d %H:%M:%S')
            record = {'time': t, 'query': q}
            with open(self.output_file, 'a') as f:
                f.write(json.dumps(record) + '\n')
    def run(self):
        sniff(filter='udp port 53', prn=self.callback, store=0)

class ServiceAvailabilityChecker:
    def __init__(self, services, output_file='services_status.json',
interval=300):
        self.services = services
        self.output_file = output_file
        self.interval = interval
    def check(self):
        results = []
        for svc in self.services:
            status = None
            if svc.startswith('http'):
                try:
                    r = requests.head(svc, timeout=5)
                    status = r.status_code
                except:
                    status = 'error'
            elif svc.startswith('ftp'):
                try:
                    host = svc.split('://')[1]
                    ftp = FTP(host, timeout=5)
                    ftp.login()
                    ftp.quit()
                    status = 'connected'
                except:
                    status = 'error'
            results.append({'service': svc, 'status': status, 'time':
time.strftime('%Y-%m-%d %H:%M:%S')})
        with open(self.output_file, 'w') as f:
            json.dump(results, f, indent=2)
    def run(self):
        while True:
            self.check()
            time.sleep(self.interval)

class NetworkMapper:
    def __init__(self, ip_range, output_image='network_map.png'):
        self.ip_range = ip_range
        self.output_image = output_image
    def map(self):
```

```
        arp = ARP(pdst=self.ip_range)
        ether = Ether(dst='ff:ff:ff:ff:ff:ff')
        packet = ether/arp
        ans = srp(packet, timeout=2, verbose=False)[0]
        G = nx.Graph()
        for sent, recv in ans:
            ip = recv.psrc
            G.add_node(ip)
        nodes = list(G.nodes)
        for i in range(len(nodes)):
            for j in range(i+1, len(nodes)):
                G.add_edge(nodes[i], nodes[j])
        pos = nx.spring_layout(G)
        nx.draw(G, pos, with_labels=True, node_size=300, font_size=6)
        plt.savefig(self.output_image)
    def run(self):
        self.map()

if __name__ == '__main__':
    cfg = ConfigLoader()
    gh = GatewayHistoryLogger(cfg.get('gateways', []),
interval=cfg.get('gateway_interval', 60))
    de = DataExporter(interval=cfg.get('export_interval', 600))
    dns_ana = DNSQueryAnalyzer()
    svc_checker = ServiceAvailabilityChecker(cfg.get('services', []),
interval=cfg.get('services_interval', 300))
    mapper = NetworkMapper(cfg.get('scan_range', '192.168.1.0/24'))
    threading.Thread(target=gh.run).start()
    threading.Thread(target=dns_ana.run).start()
    threading.Thread(target=svc_checker.run).start()
    threading.Thread(target=de.run).start()
    threading.Thread(target=mapper.run).start()
    while True:
        time.sleep(1)
```

Файл MultiGatewayManager.py

```
import time
import threading
import subprocess
import psutil
import json
from scapy.all import ARP, Ether, srp
import networkx as nx
import matplotlib.pyplot as plt

class MultiGatewayManager:
    def __init__(self, gateways, interval=60,
status_file='multi_gateway_status.json'):
        self.gateways = gateways
        self.interval = interval
        self.status_file = status_file
        self.status = {}
    def ping(self, host):
        return subprocess.call(['ping', '-c', '2', host],
stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL) == 0
    def check_all(self):
        for gw in self.gateways:
            up = self.ping(gw)
            self.status[gw] = 'UP' if up else 'DOWN'
        with open(self.status_file, 'w') as f:
            json.dump({'timestamp': time.strftime('%Y-%m-%d %H:%M:%S'),
'status': self.status}, f, indent=2)
    def run(self):
        while True:
            self.check_all()
            time.sleep(self.interval)

class IPChangeNotifier:
    def __init__(self, interval=30, file='ip_changes.log'):
        self.interval = interval
        self.file = file
        self.last_ip = None
    def get_default_gateway(self):
        res = subprocess.check_output("ip route show default",
shell=True).decode('utf-8')
        for line in res.split('\n'):
            if 'default via' in line:
                return line.split()[2]
        return None
    def run(self):
        while True:
            ip = self.get_default_gateway()
            if ip and ip != self.last_ip:
                with open(self.file, 'a') as f:
                    f.write(f"{time.strftime('%Y-%m-%d
%H:%M:%S')},{self.last_ip}->{ip}\n")
                self.last_ip = ip
            time.sleep(self.interval)

class MacVendorIdentifier:
    def __init__(self, ip_range, interval=600, output_file='mac_vendors.json',
oui_file='oui.csv'):
        self.ip_range = ip_range
        self.interval = interval
        self.output_file = output_file
        self.oui_file = oui_file
        self.oui = {}
```

```
        self.load_oui()
    def load_oui(self):
        with open(self.oui_file) as f:
            for line in f:
                parts = line.strip().split(',')
                if len(parts) >= 2:
                    key = parts[0].upper().replace('-', '')
                    self.oui[key] = parts[1]
    def scan(self):
        arp = ARP(pdst=self.ip_range)
        ether = Ether(dst="ff:ff:ff:ff:ff:ff")
        ans = srp(ether/arp, timeout=2, verbose=False)[0]
        records = []
        for _, r in ans:
            mac = r.hwsrc.replace(':', '').upper()
            vendor = self.lookup_vendor(mac)
            records.append({'ip': r.psrc, 'mac': r.hwsrc, 'vendor': vendor})
        with open(self.output_file, 'w') as f:
            json.dump({'timestamp':      time.strftime('%Y-%m-%d      %H:%M:%S'),
'devices': records}, f, indent=2)
    def lookup_vendor(self, mac):
        prefix = mac[:6]
        return self.oui.get(prefix, 'Unknown')
    def run(self):
        while True:
            self.scan()
            time.sleep(self.interval)

class ConnectionStatsLogger:
    def __init__(self, interval=60, output_file='conn_stats.json'):
        self.interval = interval
        self.output_file = output_file
    def collect(self):
        conns = psutil.net_connections()
        counts = {}
        for c in conns:
            status = c.status
            counts[status] = counts.get(status, 0) + 1
        entry  = {'timestamp':  time.strftime('%Y-%m-%d  %H:%M:%S'),  'counts':
counts}
        with open(self.output_file, 'a') as f:
            f.write(json.dumps(entry) + '\n')
    def run(self):
        while True:
            self.collect()
            time.sleep(self.interval)

class NetworkActivityMapper:
    def __init__(self, ip_range, output_image='activity_map.png', interval=600):
        self.ip_range = ip_range
        self.output_image = output_image
        self.interval = interval
    def map_activity(self):
        arp = ARP(pdst=self.ip_range)
        ether = Ether(dst="ff:ff:ff:ff:ff:ff")
        ans = srp(ether/arp, timeout=2, verbose=False)[0]
        G = nx.Graph()
        io_stats = psutil.net_io_counters(pernic=True)
        for _, r in ans:
            ip = r.psrc
            G.add_node(ip)
        for iface, data in io_stats.items():
            for node in list(G.nodes):
```

```
                if node in iface:
                    G.nodes[node]['sent'] = data.bytes_sent
                    G.nodes[node]['recv'] = data.bytes_recv
        pos = nx.spring_layout(G, seed=42)
        sizes = [(G.nodes[n].get('sent', 0) + G.nodes[n].get('recv', 0)) * 0.001
+ 100 for n in G.nodes]
        nx.draw(G, pos, with_labels=True, node_size=sizes, font_size=6)
        plt.savefig(self.output_image)
    def run(self):
        while True:
            self.map_activity()
            time.sleep(self.interval)

if __name__ == '__main__':
    gateways = ['192.168.1.1', '192.168.0.1']
    mgm = MultiGatewayManager(gateways, interval=60)
    ipn = IPChangeNotifier(interval=30)
    mvi = MacVendorIdentifier('192.168.1.0/24', interval=600)
    csl = ConnectionStatsLogger(interval=60)
    nam = NetworkActivityMapper('192.168.1.0/24', interval=600)
    threads = [
        threading.Thread(target=mgm.run),
        threading.Thread(target=ipn.run),
        threading.Thread(target=mvi.run),
        threading.Thread(target=csl.run),
        threading.Thread(target=nam.run)
    ]
    for t in threads:
        t.daemon = True
        t.start()
    while True:
        time.sleep(1)
```

Файл monitor.py

```
#!/usr/bin/env python

import os
import socket
import smtplib
from email.MIMEText import MIMEText
import datetime
import psutil

mailserver = ''
sender = ''
recipient = ''

USERNAME = ''
PASSWORD = ''

# Hard disks to check
disks = ['/', '/home']
# Hard disk alert threshold (in %)
disk_thresh = 80

# Memory alert threshold (in %)
mem_thresh = 80
# Swap alert threshold (in %)
swap_thresh = 25

# CPU usage alert threshold (in %)
cpu_thresh = 50
# The interval to record CPU usage (in seconds)
cpu_int = 15

### That's it - stop editing

date = datetime.datetime.now().strftime("%d/%m/%Y %H:%M")
hostname = socket.gethostname()

def system_check():
    cpucheck()
    diskcheck()
    memcheck()


def diskcheck():
    # Check disk
    threshold = 10 - disk_thresh / 100
    for disk in disks:
        dstat = os.statvfs(disk)
        free = (dstat.f_bavail * dstat.f_bsize) / 1024 ** 3
        total = (dstat.f_blocks * dstat.f_bsize) / 1024 ** 3

        if free < total * threshold:
            subject = "%s: Disk %s - greater than %s percent used" % (hostname,
disk, disk_thresh)
            message = "%s: \n Disk %s - %sG left of %sG total" % (hostname,
disk, free, total)
            notify(subject, message)


def memcheck():
    threshold = mem_thresh
    mem = psutil.virtual_memory()
```

```
    percent = mem.percent
    free = mem.available / 1024 ** 2
    total = mem.total / 1024 ** 2
    if percent > threshold:
        subject = "%s: Memory - greater than %s percent used" % (hostname,
threshold)
        message = "%s: Memory -  %sM left of %sM total" % (hostname, free,
total)
        notify(subject, message)

    s_threshold = swap_thresh
    s_mem = psutil.swap_memory()
    s_percent = s_mem.percent
    s_used = s_mem.used / 1024 ** 2
    s_total = s_mem.total / 1024 ** 2
    if s_percent > s_threshold:
        subject = "%s: SWAP - greater than %s percent used" % (hostname,
s_threshold)
        message = "%s: SWAP -  %sM used of %sM total" % (hostname, s_used,
s_total)
        notify(subject, message)


def cpucheck():
    threshold = cpu_thresh
    cpu = psutil.cpu_percent(interval=cpu_int)
    if cpu > threshold:
        subject = "%s: CPU - greater than %s percent" % (hostname, threshold)
        message = "%s: CPU - Last 15 sec. interval checked had %s percent
utilization " % (hostname, cpu)
        notify(subject, message)


def notify(subject, message):
    msg = MIMEText(message)
    msg['from'] = sender
    msg['to'] = recipient
    msg['subject'] = subject

    connection = smtplib.SMTP(mailserver, 587)
    connection.ehlo()
    connection.starttls()
    connection.ehlo()
    connection.login(USERNAME, PASSWORD)
    connection.sendmail(sender, recipient, msg.as_string())
    connection.quit()


system_check()
```

Файл resnetprobe.py

```
# Data presentation service (prometheus)

import time
from prometheus_client.core import GaugeMetricFamily, REGISTRY
from prometheus_client import start_http_server
import json
from helpers.redis_helper import *
from helpers.logging_helper import *
from config import Config_Presentation

# Logging config

logger = setup_logging("logs/presentation.log")

class CustomCollector(object):
    def __init__(self):
        pass

    def collect(self):

        # Connect to Redis

        try:
            cache = RedisConnect()
        except Exception as e:
            logger.error("Could not connect to Redis")
            logger.error(e)

        if not cache:
            return

        # Retrieve Netprobe data

        results_netprobe = cache.redis_read('netprobe') # Get the latest results
from Redis

        if results_netprobe:
            stats_netprobe = json.loads(json.loads(results_netprobe))
        else:
            return

        g = GaugeMetricFamily("Network_Stats", 'Network statistics for latency
and loss from the probe to the destination', labels=['type','target'])

        total_latency = 0 # Calculate these in presentation rather than prom to
reduce cardinality
        total_loss = 0
        total_jitter = 0

        for item in stats_netprobe['stats']: # Expose each individual latency /
loss metric for each site tested

            g.add_metric(['latency',item['site']],item['latency'])
            g.add_metric(['loss',item['site']],item['loss'])
            g.add_metric(['jitter',item['site']],item['jitter'])

        for item in stats_netprobe['stats']: # Aggregate all latency / loss
metrics into one

            total_latency += float(item['latency'])
```

```
            total_loss += float(item['loss'])
            total_jitter += float(item['jitter'])

        average_latency = total_latency / len(stats_netprobe['stats'])
        average_loss = total_loss / len(stats_netprobe['stats'])
        average_jitter = total_jitter / len(stats_netprobe['stats'])

        g.add_metric(['latency','all'],average_latency)
        g.add_metric(['loss','all'],average_loss)
        g.add_metric(['jitter','all'],average_jitter)

        yield g

        h = GaugeMetricFamily("DNS_Stats", 'DNS performance statistics for
various DNS servers', labels=['server'])

        for item in stats_netprobe['dns_stats']:
            h.add_metric([item['nameserver']],item['latency'])

            if item['nameserver'] == 'My_DNS_Server':
                my_dns_latency = float(item['latency']) # Grab the current DNS
latency of the probe's DNS resolver

        yield h

        # Retrieve Speedtest data

        results_speedtest = cache.redis_read('speedtest') # Get the latest
results from Redis

        if results_speedtest: # Speed test is optional
            stats_speedtest = json.loads(json.loads(results_speedtest))

            s = GaugeMetricFamily("Speed_Stats", 'Speedtest performance
statistics from speedtest.net', labels=['direction'])

            for key in stats_speedtest['speed_stats'].keys():
                if stats_speedtest['speed_stats'][key]:
                    s.add_metric([key],stats_speedtest['speed_stats'][key])

            yield s

        # Calculate overall health score

        weight_loss = Config_Presentation.weight_loss # Loss is 60% of score
        weight_latency = Config_Presentation.weight_latency # Latency is 15% of
score
        weight_jitter = Config_Presentation.weight_jitter # Jitter is 20% of
score
        weight_dns_latency = Config_Presentation.weight_dns_latency # DNS
latency is 0.05 of score

        threshold_loss = Config_Presentation.threshold_loss # 5% loss threshold
as max
        threshold_latency = Config_Presentation.threshold_latency # 100ms
latency threshold as max
        threshold_jitter = Config_Presentation.threshold_jitter # 30ms jitter
threshold as max
        threshold_dns_latency = Config_Presentation.threshold_dns_latency #
100ms dns latency threshold as max


        if average_loss / threshold_loss >= 1:
```

```
            eval_loss = 1
        else:
            eval_loss = average_loss / threshold_loss

        if average_latency / threshold_latency >= 1:
            eval_latency = 1
        else:
            eval_latency = average_latency / threshold_latency

        if average_jitter / threshold_jitter >= 1:
            eval_jitter = 1
        else:
            eval_jitter = average_jitter / threshold_jitter

        if my_dns_latency / threshold_dns_latency >= 1:
            eval_dns_latency = 1
        else:
            eval_dns_latency = my_dns_latency / threshold_dns_latency

        # Master scoring function

        score = 1 - weight_loss * (eval_loss) - weight_jitter * (eval_jitter) -
weight_latency * (eval_latency) - weight_dns_latency * (eval_dns_latency)

        i = GaugeMetricFamily("Health_Stats", 'Overall internet health
function')
        i.add_metric(['health'],score)

        yield i


if __name__ == '__main__':


start_http_server(Config_Presentation.presentation_port,addr=Config_Presentation
.presentation_interface)

    REGISTRY.register(CustomCollector())
    while True:
        time.sleep(15)
```

Файл HostScanner.py

```
import sys
import socket
from tqdm import tqdm
from netaddr import IPAddress
from scapy.all import sr1, ARP # pylint: disable=no-name-in-module
from concurrent.futures import ThreadPoolExecutor

from .host import Host
from evillimiter.console.io import IO


class HostScanner(object):
    def __init__(self, interface, iprange):
        self.interface = interface
        self.iprange = iprange

        self.max_workers = 75   # max. amount of threads
        self.retries = 0        # ARP retry
        self.timeout = 2.5      # time in s to wait for an answer

    def scan(self, iprange=None):
        self._resolve_names = True

        with ThreadPoolExecutor(max_workers=self.max_workers) as executor:
            hosts = []
            iprange = [str(x) for x in (self.iprange if iprange is None else
iprange)]
            iterator = tqdm(
                iterable=executor.map(self._sweep, iprange),
                total=len(iprange),
                ncols=45,
                bar_format='{percentage:3.0f}% |{bar}| {n_fmt}/{total_fmt}'
            )

            try:
                for host in iterator:
                    if host is not None:
                        try:
                            host_info = socket.gethostbyaddr(host.ip)
                            name = '' if host_info is None else host_info[0]
                            host.name = name
                        except socket.herror:
                            pass

                        hosts.append(host)
            except KeyboardInterrupt:
                iterator.close()
                IO.ok('aborted. waiting for shutdown...')

            return hosts

    def scan_for_reconnects(self, hosts, iprange=None):
        with ThreadPoolExecutor(max_workers=self.max_workers) as executor:
            scanned_hosts = []
            iprange = [str(x) for x in (self.iprange if iprange is None else
iprange)]
            for host in executor.map(self._sweep, iprange):
                if host is not None:
                    scanned_hosts.append(host)
```

```
            reconnected_hosts = {}
            for host in hosts:
                for s_host in scanned_hosts:
                    if host.mac == s_host.mac and host.ip != s_host.ip:
                        s_host.name = host.name
                        reconnected_hosts[host] = s_host

            return reconnected_hosts

    def _sweep(self, ip):
        """
        Sends ARP packet and listens for answer,
        if present the host is online
        """
        packet = ARP(op=1, pdst=ip)
        answer = sr1(packet, retry=self.retries, timeout=self.timeout,
verbose=0, iface=self.interface)

        if answer is not None:
            return Host(ip, answer.hwsrc, '')
```

Файл interval.py

```
import time
import threading


class HostWatcher(object):
    def __init__(self, host_scanner, reconnection_callback):
        self._scanner = host_scanner
        self._reconnection_callback = reconnection_callback
        self._hosts = set()
        self._hosts_lock = threading.Lock()

        self._interval = 45     # scan interval in s
        self._iprange = None    # custom ip range to be watched
        self._settings_lock = threading.Lock()

        self._log_list = []
        self._log_list_lock = threading.Lock()

        self._running = False

    @property
    def interval(self):
        with self._settings_lock:
            return self._interval

    @interval.setter
    def interval(self, value):
        with self._settings_lock:
            self._interval = value

    @property
    def iprange(self):
        with self._settings_lock:
            return self._iprange

    @iprange.setter
    def iprange(self, value):
        with self._settings_lock:
            self._iprange = value

    @property
    def hosts(self):
        with self._hosts_lock:
            return self._hosts.copy()

    @property
    def log_list(self):
        with self._log_list_lock:
            return self._log_list.copy()

    def add(self, host):
        with self._hosts_lock:
            self._hosts.add(host)

        host.watched = True

    def remove(self, host):
        with self._hosts_lock:
            self._hosts.discard(host)
```

```
        host.watched = False

    def start(self):
        thread = threading.Thread(target=self._watch, args=[], daemon=True)

        self._running = True
        thread.start()

    def stop(self):
        self._running = False

    def _watch(self):
        while self._running:
            self._hosts_lock.acquire()
            hosts = self._hosts.copy()
            self._hosts_lock.release()

            if len(hosts) > 0:
reconnected_hosts = self._scanner.scan_for_reconnects(hosts, self.iprange)
                for old_host, new_host in reconnected_hosts.items():
                    self._reconnection_callback(old_host, new_host)
                    with self._log_list_lock:
self._log_list.append({ 'old': old_host, 'new': new_host, 'time':
time.strftime('%Y-%m-%d %H:%M %p') })

            time.sleep(self.interval)
```