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 доступу у мережу на основі технології Cisco NAC”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Ожеховський В.В.
«____» _____ 2025 р.

Керівник проекту
доктор технічних наук, професор
_____ Коваленко О.В.
«____» _____ 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

Ожеховському Владиславу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи контролю доступу у мережу на основі технології Cisco NAC*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 46-02 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 доступу у мережу на основі технології Cisco NAC*

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
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Ожеховський В.В.
(прізвище та ініціали)

АНОТАЦІЯ

Ожеховський В.В. Програмне забезпечення системи контролю доступу у мережу на основі технології Cisco NAC. 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 доступу у мережу на основі технології Cisco NAC.

Метою розробки є програмне забезпечення системи контролю доступу у мережу на основі технології Cisco NAC.

Результат роботи – програмна реалізація системи контролю доступу у мережу на основі технології Cisco NAC.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, контроль доступу у мережу, Cisco NAC

ABSTRACT

Ozhehovskii V.V. Software for a network access control system based on Cisco NAC technology. 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 a network access control system based on Cisco NAC technology.

The purpose of the development is software for a network access control system based on Cisco NAC technology.

The result of the work is a software implementation of a network access control system based on Cisco NAC technology.

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

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

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

The program was developed in the Python environment.

Keywords: computer engineering, network access control, Cisco NAC

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЗІ	–	захист інформації
ІБ	–	інформаційна безпека
ІТ	–	інформаційна технологія
КІС	–	корпоративні інформаційні системи
ЛОМ	–	локальна обчислювальна мережа
НСД	–	несанкціонований доступ
ОС	–	операційна система
СГ КІС	–	сегмент КІС
СЗІ	–	система захисту інформації
IPS	–	система запобігання вторгнень
NAC	–	Network Admission Control
NIDS	–	система виявлення мережних вторгнень

ВСТУП

Актуальність теми. Контроль доступу в мережу (Network Admission Control, NAC) – це набір технологій і рішень, фундаментом яких служить загальногалузева ініціатива, реалізована під патронажем Cisco Systems. NAC використовує інфраструктуру мережі для контролю над дотриманням політики безпеки на всіх пристроях, що прагнуть одержати доступ до ресурсів мережі. Цим шляхом знижується збиток, що можуть заподіяти виникаючі погрози безпеки. Використовуючи NAC, клієнти одержують можливість надавати мережний доступ тільки дотримує запропоновані вимоги, безпечним кінцевим пристроям (наприклад, комп'ютерам, серверам і КПК) і обмежувати доступ для пристроїв, що не відповідають вимогам. Такі можливості реалізовані двома шляхами:

– Пристрій контролю доступу в мережу (NAC Appliance technology) на базі лінійки продуктів Cisco Clean Access, забезпечує швидке розгортання із сервісами автономної експертизи на кінцевих вузлах, керування політиками й коррективні дії.

– Архітектура контролю доступу в мережу (NAC Framework technology), реалізована через програму Cisco контролю доступу в мережу (Cisco Network Admission Control Program), поєднує інтелектуальну мережну інфраструктуру з рішеннями більш ніж 75 розроблювачів провідних антивірусних програм і іншого програмного забезпечення в області безпеки й керування.

Мережна стратегія Cisco дозволяє використовувати наявні інвестиції для рішення сучасних невідкладних завдань безпеки, надаючи архітектурну платформу, що розвивається, для що попереджає автоматизованого керування погрозами в реальному масштабі часу.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи контролю доступу у мережу на основі технології Cisco NAC.

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

– Огляд існуючих систем контролю доступу у мережу на основі технології Cisco NAC.

– Дослідження системи контролю доступу у мережу на основі технології Cisco NAC.

– Програмна реалізація системи контролю доступу у мережу на основі технології Cisco NAC.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі контролю доступу у мережу на основі технології Cisco NAC.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи контролю доступу у мережу на основі технології Cisco NAC, 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

Система призначена для реалізації програмного забезпечення контролю доступу у мережу на основі технології Cisco NAC**.** Реалізуються наступні переваги.

Істотне підвищення рівня безпеки:

– Забезпечується дотримання політики безпеки на кінцевих вузлах (ПК, КПК, серверах).

– Діє превентивний захист від хробаків, вірусів, шпигунських програм і злочинних програм. Основна увага приділяється профілактичним мірам, а не відповідним діям.

Розширення існуючих інвестицій:

– Широка інтеграція із програмним забезпеченням різних розроблювачів в області безпеки й керування.

– Ріст ефективності інвестицій у мережну інфраструктуру й стороннє програмне забезпечення.

Більше стабільна робота компаній:

– Повномасштабне керування доступом, включаючи всі методи.

– Виключено вплив не відповідають вимогам безпеки й заражених кінцевих вузлів на роботу мережі.

– Знижуються операційні витрати, пов'язані з виявленням і коректуванням не відповідають вимогам безпеки, заражених і інфікованих систем.

Повномасштабний контроль:

– Експертиза всіх кінцевих точок і охват всіх методів доступу, включаючи LAN, бездротовий, дистанційний доступ і WAN.

1.2 Область застосування

Областю застосування є забезпечення контролю доступу у мережу на основі технології Cisco NAC. Віруси, хробаки й шпигунські програми, що загрожують безпеки мереж, завдають шкоди клієнтам і віднімають у компаній гроші, виробничі ресурси й вигідні можливості. Повсюдне поширення мобільних комп'ютерів тільки збільшило масштаб цієї погрози. Мобільні користувачі можуть підключатися до мережі Інтернет і до офісних мереж, перебуваючи вдома або в громадських місцях. Вони легко й найчастіше неусвідомлено підхоплюють віруси й переносять їх у корпоративне робітниче середовище, заражаючи всю мережу.

Контроль доступу в мережу (Network Admission Control, NAC) був спеціально розроблений для того, щоб забезпечити достатній захист всіх кінцевих пристроїв (ПК, лептопів, серверів, смартфонів і КПК), що одержують доступ до мережних ресурсів, від погроз безпеки, які присутні в мережі. Передові рішення по NAC прийняті на озброєння провідними розроблювачами антивірусних, програм керування й забезпечення безпеки й залучають інтерес преси, аналітиків і компаній самого різного розміру.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи контролю доступу у мережу на основі технології Cisco NAC, 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

Система керування мережним доступом – Network Access Control (NAC) – становить один із шарів забезпечення безпеки й цілісності мережі, додатків і даних. Призначення NAC складається у виявленні й перевірці «благонадійності» кожного пристрою в мережі. При виявленні пристрою система перевіряє його на відповідність правилам, установленим адміністратором, оцінюючи ймовірність «добропорядного поводження» цього пристрою. Правила звичайно вимагають наявності на кінцевих пристроях мінімального програмного забезпечення (наприклад, антивірусного пакета).

Всі продукти, розглянуті в рамках даного порівняльного аналізу, а саме EndForce Enterprise від Sophos, Dynamic NAC for Windows від InfoExpress, Policy Enforcer від McAfee і Safe Access від StillSecure, передбачають захист від погроз, що виходять від кінцевих пристроїв, що підключаються до локальної мережі. Всі продукти є програмними пакетами, установлюваними на власних апаратних засобах користувача. Альтернативний варіант – апаратні пристрої NAC, тема, що заслуговує окремого порівняльного аналізу. Ще один клас продуктів, які нерідко ставлять в один ряд з мережними шлюзами, припускає виконання перевірки до з'єднання (pre-connect) і забезпечує фільтрацію й перевірку «благонадійності» трафіку, ініціюємого ззовні щодо локальної мережі.

Методи контролю дотримання політики безпеки

Існує кілька розповсюджених методів приведення NAC у виконання. Метод з використанням агента припускає функціонування на кожній системі програмного пакета, що виконує перевірку цієї системи й у випадку

недотримання вимог безпеки обмежуючий її доступ до мережних ресурсів. Метод з використанням протоколу динамічної конфігурації DHCP передбачає призначення системам, що не задовольняють вимогам політики безпеки, мережної конфігурації, що обмежує їхня здатність взаємодії з іншими системами. Метод на основі простого протоколу керування мережею SNMP використовує мережні комутатори з можливістю організації SNMP-керованої віртуальної мережі VLAN; при цьому кінцеві пристрої, що не відповідають вимогам безпеки, переводяться в VLAN з обмеженим доступом. Нарешті, метод на основі протоколу 802.1x, припускає використання комутаторів з підтримкою стандарту 802.1x; клієнт, що активує порт комутатора, щораз прикріплюється до VLAN з обмеженим доступом, доти поки не буде пізнаний і перевірений сервером NAC.

Один з об'єктів нашого порівняльного аналізу – продукт Dynamic NAC for Windows від InfoExpress – передбачає використання ще одного методу контролю дотримання політики безпеки з переадресацією протоколу дозволу адрес Address Resolution Protocol (ARP). Типи реалізованих методів контролю розрізняються залежно від виконання перевірки перед з'єднанням (pre-connect) або після з'єднання (post-connect). Метод на основі 802.1x ставиться до типу pre-connect, оскільки трафік, що виходить від нового кінцевого пристрою, не допускається в мережу, поки не пройде перевірку на благонадійність. Інші методи в цілому ставляться до типу post-connect, і в них є свої слабкі місця.

Кожний метод контролю має позитивні й негативні сторони. Агентський контроль (не плутати з агентською перевіркою) погано захищає системи, що не використовують агент. DHCP-контроль – невдалий варіант для систем зі статичними IP-адресами. Метод на основі SNMP і 802.1x припускає використання апаратних пристроїв, якими багато організацій не розташовують.

Sophos EndForce Enterprise

EndForce Enterprise (EE) – NAC-рішення на основі сервера Windows, що передбачає виконання перевірки як до, так і після з'єднання. Sophos передбачає

планову допомогу в установці продукту новим клієнтам на місці, але я виконав установку, скориставшись підтримкою по телефону.

Архітектура. EE реалізує архітектуру «клієнтський агент/сервер» з підтримкою контролю виконання вимог безпеки на агенті EndForce Agent, комутаторах 802.1x, серверах Microsoft або Lucent DHCP і VPN-концентраторах. Також передбачена підтримка інфраструктури Cisco NAC. У великих мережах EE дозволяє встановлювати кілька серверів додатка з аналогічною конфігурацією для реалізації схеми балансування мережного навантаження Network Load Balancing (NLB).

У всіх варіантах реалізації контролю EE припускає використання агента, установлюваного на кінцевому пристрої, для перевірки відповідності вимогам політики безпеки. EE включають клієнти на основі Active і служб Windows, але не включають клієнти для систем Linux або Macintosh. Перед установкою агента необхідно створити спеціальний настановний файл MSI для завдання IP-адреси сервера додатка EE, з яким він буде працювати, потім вибрати один із трьох режимів роботи агента: Quarantine – перевірка клієнта, виконувана до його допуску в мережу, а також згодом, із приміщенням у карантин, якщо виявляється порушення вимог політики безпеки; Continuous – аналогічно варіанту Quarantine, але без ізоляції клієнта при виявленні порушення вимог політики безпеки; On Demand – варіант, призначений для VPN-додатків.

На відміну від інших учасників даного порівняльного аналізу, EE припускає орієнтацію на кінцевого користувача (замість орієнтації на систему) у підході до реалізації контролю дотримання політики NAC. При використанні EE кінцеві пристрої перебувають в одному із трьох станів: відомий користувач на керованому кінцевому пристрої, відомий користувач на некерованому кінцевому пристрої й невідомому користувачі на невизначеному кінцевому пристрої. Реалізована EE керуюча програма Policy Manager дозволяє призначати політики для користувальницьких груп EE, які можна асоціювати з користувальницькими групами Active Directory (AD).

Користувачі часто застосовують DHCP-метод контролю дотримання вимог безпеки (для приміщення в карантин нових клієнтських DHCP-систем до виконання їхньої перевірки) і агентський метод (для постійного керування й періодичної перевірки систем компанії). EE реалізує DHCP-контроль із використанням модуля DHCP Enforcer, установлюваного на сервері DHCP. У комбінації із застосуванням класів користувачів DHCP це дозволяє EE змушувати DHCP-сервер призначати кінцевим пристроям, що невдало пройшли перевірку, мережні адреси, що обмежують їхній доступ до мережних ресурсів. Наприклад, кінцевий пристрій, що порушує вимоги політики безпеки, може одержати IP-адресу, маску підмережі й адресу мережного шлюзу, що відкривають йому доступ тільки до сервера виправлення (remediation server).

Установка. EE функціонує під Windows Server 2012 з конфігурацією, побудованою з використанням Microsoft IIS і Internet Authentication Service (IAS). Продукт також вимагає наявності системи Microsoft SQL Server 2014. Основна процедура установки на сервері EE пройшла цілком гладко, після чого протягом години виконувався процес побудови конфігурації за участю IIS і IAS, а потім – через Web-інтерфейс EE – налаштування агентського пакета MSI. Web-консоль передбачає появу спливаючих вікон, тому мені довелося виключити блокування спливаючих вікон на моєму комп'ютері. Я створив політику з вимогою використання тільки одного агента EE; ця політика стала використовуваною за замовчуванням, оскільки була створена першою. Потім я створив користувальницьку групу EE і асоціював її із групою доменних користувачів, після чого призначив тільки що створену політику цій групі користувачів EE.

Щоб почати тестування, я встановив агент на робочій станції Windows, після чого з'ясувалося, що користувачі повинні вказати агентові свої ID і паролі для реєстрації на сервері EE. Для автентифікації й реєстрації клієнтів на сервері EE агент використовує IAS. Спершу автентифікація закінчилася невдачею, оскільки в мого ID були відсутні привілею віддаленого доступу, тому в IAS я створив політику віддаленого доступу, що ігнорує параметри облікового запису

користувача, що стосуються підключення. Після цього реєстрація пройшла успішно, і агент виконав завантаження політик за замовчуванням. До цього моменту робоча станція перебувала в карантині, оскільки я виконав відповідне налаштування в настановному файлі агента .msi. Потім я змінив налаштування політики й увів вимогу наявності антивірусного пакета, який у мене не було. Незабаром після цього – у межах заданого інтервалу відновлення політики – робоча станція була знову поміщена в карантин. Я спробував установити агент на другу робочу станцію, і система знову відхилила спробу мережного доступу. Web-консоль ЇЇ видала інформацію про приміщення в карантин із вказівкою причини, як показано на рисунку 2.1.

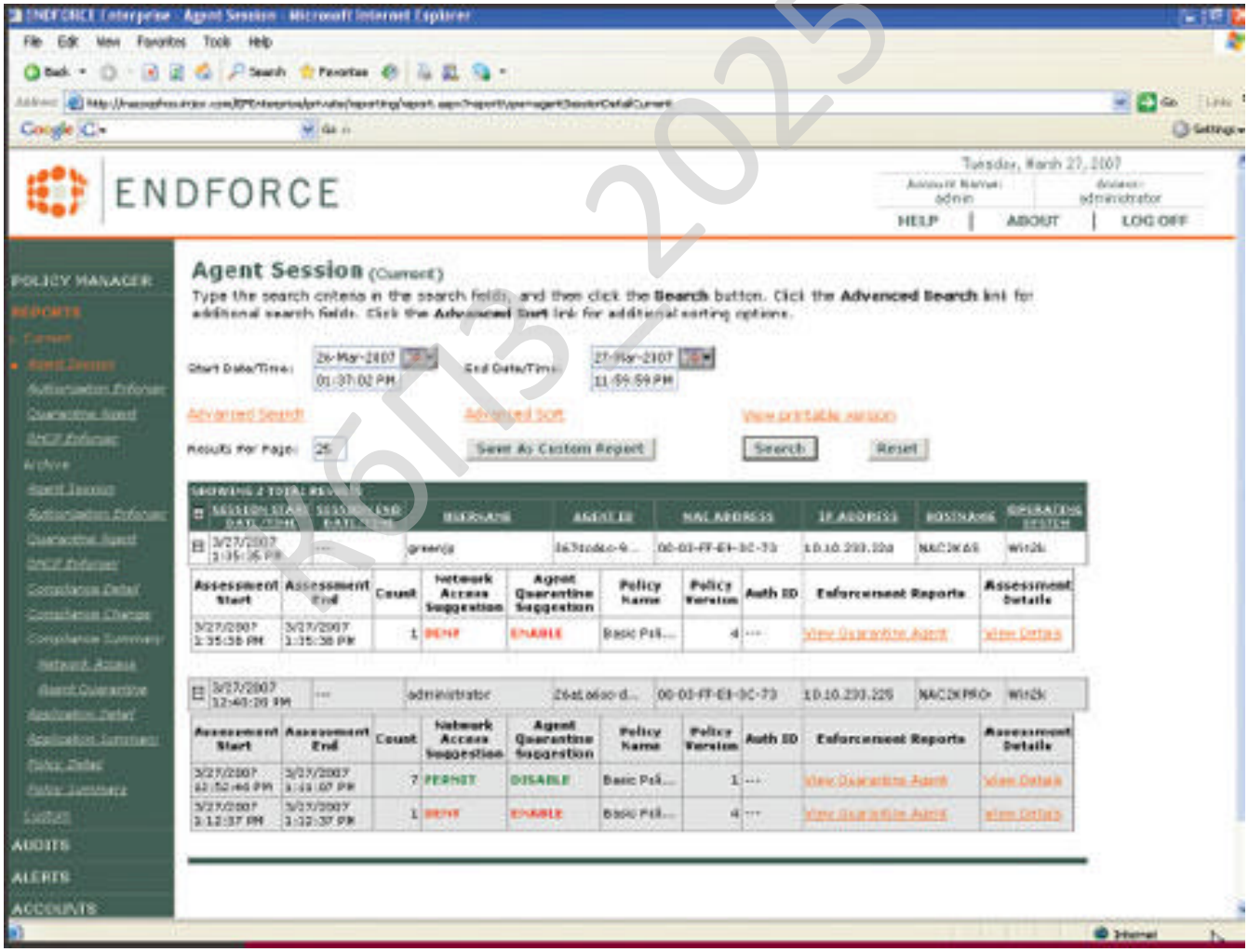

Рисунок 2.1 – Інтерфейс користувача Sophos EndForce Enterprise

Висновок. ЕЕ може стати ефективним доповненням до набору засобів забезпечення безпеки, причому найвищий рівень захисту реалізується з використанням апаратного пристрою 802.1x або Cisco NAC, що працює в режимі перевірки до підключення. Комбінація агентського й DHCP-методів реалізації контролю здатна забезпечити захист від найпоширеніших погроз мережної безпеки. Я вважаю структуру ЕЕ більше складної в реалізації й керуванні в порівнянні з іншими системами, а необхідність указувати агентові ID і паролі користувачів може виявитися трохи обтяжною. Орієнтація на користувача погодиться з характером керування багатьма мережами, але все-таки хотілося б, щоб на консолі був представлений список всіх виявлених кінцевих пристроїв, а не тільки тих, які мають агент або призначені DHCP-адреси. Довідкова система консолі надає опис всіх панелей конфігурації, але мені не завжди вдавалося знайти пояснення до описів. Я також шукав – і не знайшов – документацію з описом технічних деталей архітектури. Мені довелося кілька разів звертатися по телефоні за допомогою до консультанта зі служби технічної підтримки.

InfoExpress Dynamic NAC for Windows

Технологія Dynamic NAC for Windows (DNW), що передбачає виконання перевірки після підключення, є присутнім у сімействі продуктів InfoExpress як установлюваний під Windows програмний пакет, а також як апаратний пристрій. Хоча цей продукт зветься DNW, по його інтерфейсі й модулю установки (cgsuite.exe) видно, що він являє собою функціональний набір із сімейства продуктів InfoExpress CyberGatekeeper (CG). Однак, щоб уникнути плутанини, будемо використовувати ім'я DNW.

Установка. Продукт передбачає деякі основні вимоги. Необхідна система Windows Server 2012 з конфігурацією, побудованої за допомогою IIS. Продукт використовує SQL Server і встановлює Microsoft SQL Server Desktop Engine (MSDE) 2014 у системі, що вказується адміністратором, бази даних, якщо не вдається виявити екземпляр SQL Server. Я вибрав установку за замовчуванням, що пройшла швидко й без проблем.

Архітектура. DNW – система типу «клієнтський агент/сервер» з підтримкою кінцевих мережних пристроїв Windows, Linux і Mac, хоча агенти Linux і Mac до останнього часу не підтримували набір функцій NAC. Доступний також агент Active. Використовувана на вибір програма керування звітністю поєднує реєстраційні записи агентів у базі даних і генерує звіти про діяльність. Сервер DNW покладає на обрані кінцеві агентські системи кожної підмережі функції «охоронців» (enforcer) виконання вимог політики.

Dynamic NAC використовує ARP-переадресацію. Для пояснення нагадаю деякі відомості з області побудови мереж. На Ethernet-карті комп'ютера під час її виготовлення кодується адресу Media Access Control (MAC). Для відправлення пакета на конкретний комп'ютер або шлюз у локальної підмережі комп'ютеру необхідно знати MAC-адресу цільового пристрою. ARP передає комп'ютеру необхідну MAC-адресу при спробі цього комп'ютера встановити зв'язок по конкретному IP-адресу. ARP-переадресація припускає відправлення комп'ютеру MAC-адреси системи, відмінної від тої, котрої належить зазначений IP-адресу. Використання ARP-переадресації дозволяє одному комп'ютеру контролювати доступ іншого до комп'ютерів мережі. Відзначимо, що ця технологія працює в мережах Windows, оскільки IP-стек Windows завжди приймає на обробку ARP-пакети, що надсилаються іншими. Досвідчений програміст міг би написати стек, що працює інакше. Агенти кожної підмережі «придивляються» до незвичайного поводження систем, на яких відсутній динамічний агент NAC і які не визначені в списку «благонадійних» систем для даної підмережі. Коли, що відхиляється від норми пристрій, намагається зв'язатися із системою, ця спроба виявляється невдалою, агент направляє пристрою ARP-пакети, що забезпечують його переадресацію, як правило, на сервер виправлень для виконання установки агента або проведення подальшого аналізу відповідності вимогам політики.

Операторський режим. DNW включає три графічних інтерфейси. Графічний інтерфейс CyberGatekeeper Policy Manager застосовується для створення наборів політики, використовуваних системою для оцінки стану

мережних кінцевих пристроїв.Web-інтерфейс CyberGatekeeper Reporting and Management System (CGRMS) дозволяє набудовувати й контролювати виконання політики на підмережах. Ще один Web-інтерфейс – CyberGatekeeper Server Configuration – використовується для налаштування різних аспектів конфігурації сервера DNW. У ході установки DNW призначаєте пароль для передбаченої за замовчуванням «кореневої» облікового запису. CGRMS дозволяє створювати додаткових користувачів, наділених повноваженнями змінювати конфігурацію DNW-сервера, міняти конфігурацію Dynamic NAC і становити звіти.

DNW вимагає виконання великого обсягу налаштування після установки. Наприклад, необхідно призначати контрольовані підмережі й списки доступу з маршрутизацією, що дозволяють системам, які підверглися санкціям (тобто системам, яким DNW обмежив мережний доступ), встановлювати зв'язок із серверами виправлень і інших мережних ресурсів, необхідними для усунення невідповідностей.

В основі реалізації DNW лежить використання політик. Як показано на рисунку 2.2, політики визначають умови When, вимоги й відповідні дії, застосовувані в тому випадку, якщо кінцевий пристрій не відповідає вимогам політики. Система перевіряє кінцевий пристрій на відповідність вимогам політики, якщо цей пристрій задовольняє всім умовам When. Вимоги політики вважаються невиконаними, якщо не задовольняється кожне із установлених вимог. Відповідна дія може передбачати висновок спливаючого повідомлення на клієнті. Для керованих клієнтів (тобто клієнтів, на яких функціонує агент DNW) відповідна дія може також передбачати наявність коду в цьому вікні, що спонукує агента до виконання програми, що може ініціювати установку програмного пакета. Адміністратори задають умови й вимоги, що перевіряються в режимі заздалегідь певних або налаштовуємих спеціально складених або базових тестів. Базові тести (Basic Tests) припускають перевірку виконання однієї умови, наприклад IP-адреси, виконуваного процесів або наявності конкретної операційної системи. Складені тести (Compound Tests) включають кілька базових

тестів; якщо кінцевий пристрій успішно проходить кожної з базових тестів, складений тест уважається успішно пройденим (true). InfoExpress пропонує безліч заздалегідь певних політик і періодично випускає оновлення, що завантажуються.

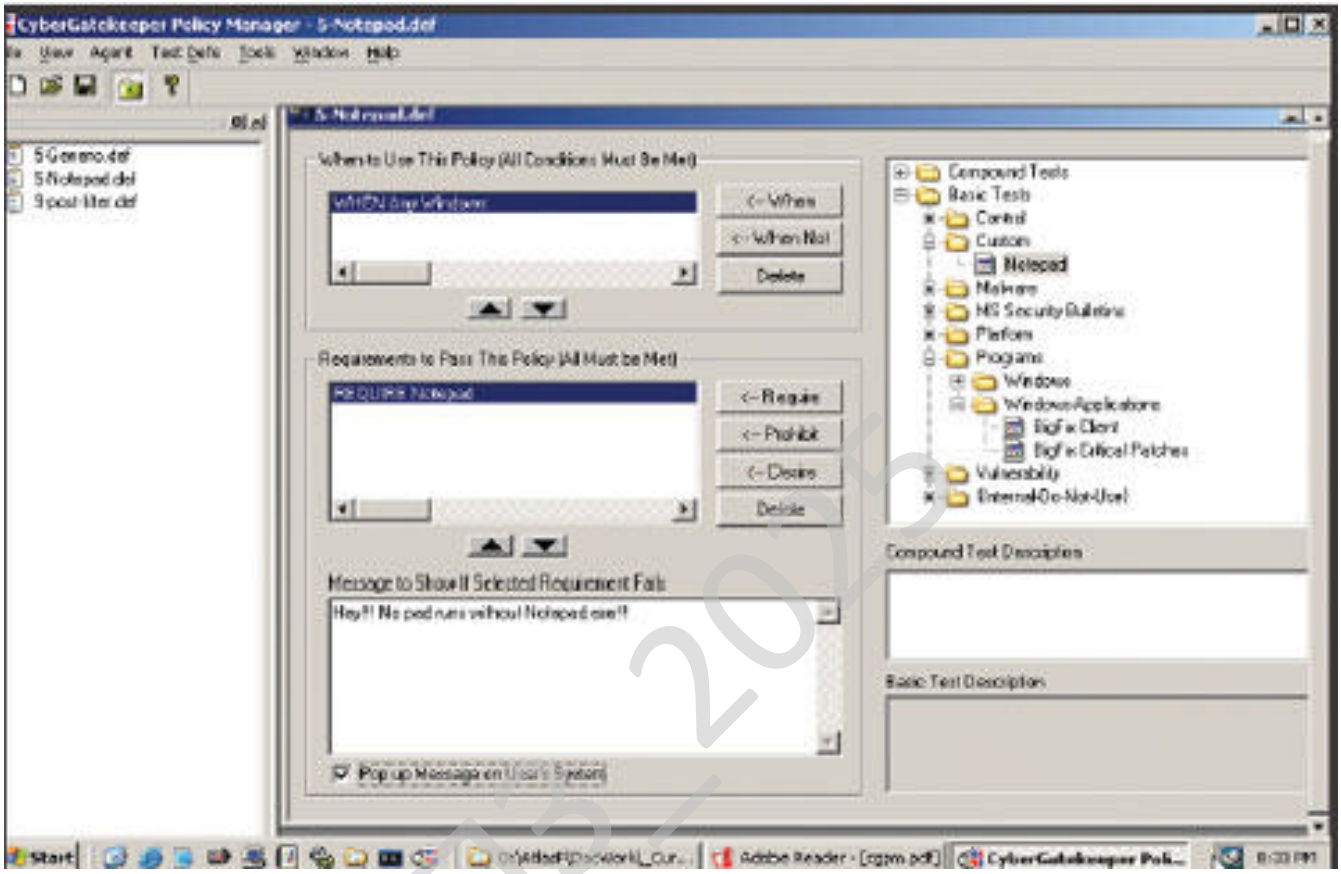

Рисунок 2.2 – Інтерфейс користувача InfoExpress Dynamic NAC for Windows

У рамках політик з використанням функції перевірки процесів (Process Tests) можна встановити вимогу наявності бажаної програми, що виконується. Щоб ускладнити завдання зловмисникові, що намагається фальсифікувати проходження тесту, DNW дозволяє перевіряти атрибути конкретних DLL, що завантажуються додатком. DNW підтримує безліч типів перевірок, включаючи перевірку версії операційної системи й мережної адреси.

Для проведення простого тесту я створив політику, відповідно до якої було потрібно функціонування Windows Notepad на кожній цільовій системі. Ще

я створив другу політику з вимогою функціонування DLL для завантажується службою черги друку. Умову When я обмежив єдиним IP-адресою, після чого завантажив політики на сервер DNW. У ході тестування з'ясувалося, що DNW застосовує до кінцевому пристрою тільки першу політику, що відповідає умові When. Мій технічний консультант сказав, що ситуація незабаром повинна змінитися, і наступні версії DNW будуть змушувати агента кінцевого пристрою застосовувати все політики, асоційовані з перевірками виконання умови When, які проходить кінцевий пристрій. Потім я створив пакет установки агента – необхідний процес попереднього завдання IP-адреси сервера DNW для агента. DNW не передбачає можливості примусової установки, тому я скористався тим же каталогом, у який DNW помістив агентський пакет, і встановив агента на двох клієнтських системах. Виявилося, що системи, що невдало пройшли перевірку на відповідність вимогам політики, не мають доступу до інших керованих систем.

Підсумок. DNW являє собою NAC-рішення, що не вимагає глибоких знань в області мережних комутаторів. Залежно від вимог і можливостей їхнього задоволення, реалізація перевірки кінцевих пристроїв з використанням заздалегідь певних тестів може виявитися більш-менш трудомісткої, а структура не здається занадто складної для розуміння. Робота DNW заснована на наявності керованих агентів у кожної підмережі, що контролюють дотримання вимог політики безпеки, однак це не створює надмірного непродуктивного навантаження на керовані системи.

McAfee Policy Enforcer

Policy Enforcer (MPE) від McAfee – програмно реалізоване NAC-рішення, що ставиться до типу post-connect (тобто з перевіркою після з'єднання), у якому використані всі можливості серверної архітектури McAfee Common Management Agent/консолі ePolicy Orchestrator (EPO). Одним з переваг MPE є здатність працювати разом з іншими продуктами McAfee із числа засобів забезпечення безпеки в рамках загального середовища керування на основі EPO.

MPE можна налаштувати на самоконтроль із централізованим агентом, а також на контроль із використанням SNTP-комутатора. MPE задіє агент, установлюваний на кінцевих пристроях Windows (клієнтах і серверах), що здійснює перевірку систем на відповідність вимогам політики. Агенти, призначувані в якості «сенсорів» (Policy Enforcer Sensors) у кожній підмережі, виявляють нові некеровані системи, прослуховуючи широкомовний трафік і DHCP-запити. Якщо в мережі є SNMP-керовані комутатори, що передбачають можливість організації VLAN, MPE віддає цим комутатором розпорядження прикріплювати нові, не минулі «перевірку на благонадійність» системи до VLAN обмеженого доступу. Агенти, призначувані в якості «сканерів» (Policy Enforcer Scanners), здійснюють перевірку систем, що не мають агента, на відповідність вимогам політики. MPE також підтримує інфраструктуру Cisco NAC.

Апаратні пристрої вселяють, що довіру мережні, і системи, що не ставляться до Windows-систем, можна помістити в список «благонадійних» вузлів Trusted Host, оскільки в противному випадку вони, що як не мають агента, не зможуть у повному обсязі проходити перевірку на відповідність вимогам політики, і MPE буде обмежувати їхній мережний доступ. Агенти Super Agents також містять копії всіх поточних наборів політик, передаючи їх кінцевим системам і зменшуючи мережний трафік на сервер EPO/MPE.

Альтернативою керованому агентові є налаштування мережі на переадресацію Web-браузерів некерованих систем на Web-сервер, з якого вони можуть здійснювати завантаження й запуск системи сканування на базі Active. Зокрема, цей метод можна використовувати для перевірки систем відвідувачів або партнерів. MPE включає код Web-сайту порталу виправлення, що полегшує створення Web-сайту виправлень і реалізацію можливостей автоматичного виконання коригувальних дій для невиконуваного кінцевим пристроєм правила.

Установка. Звичайно MPE установлюють на той же сервер, де функціонує EPO, однак цю установку можна здійснити й в іншому місці, щоб забезпечити розподіл навантаження. EPO використовує базу даних SQL Server для зберігання

конфігурації й інформації про результати перевірки клієнтів. Після установки ЕРО я встановив МРЕ і вибрав варіант, що передбачає установку порталу виправлення.

Операторський режим. На рисунку 2.3 показано консоль ЕРО. Деревоподібна схема на консолі ліворуч містить системний каталог, у якому можна створити багаторівневу ієрархічну структуру організації кінцевих систем. При виборі елемента деревоподібної схеми на дисплеї відображається відповідний екран конфігурації. Консоль зручна й проста в керуванні. Меню, що відкривається клацанням правої клавіші миші на рядку каталогу на деревоподібній схемі консолі, містить можливості, що дозволяють імпортувати системи з контейнерів AD. Якщо не задати спеціальну установку функції автоматичного імпортування, що передбачає приміщення нових систем у папки ієрархічної структури каталогів по IP-адресі, ЕРО розміщає нові системи в папці Lost&Found. Із цієї папки шляхом перетаскування можна перемістити їх у будь-яку папку на вибір. З того ж меню, що відкривається клацанням правої клавіші на папці каталогу або ім'я комп'ютера, ЕРО висилає свого агента на обрані системи. За допомогою агента ЕРО, що функціонує на обраних кінцевих пристроях, я розгорнув на цих системах сканери МРЕ Scanners через закладку Tasks (відкривається клацанням миші на папці або ім'ї комп'ютера).

Наступним кроком стала установка сенсорів МРЕ Sensors у підмережах. Це завдання я виконав з екрана, що відкривається клацанням на McAfee Policy Enforcer на деревоподібній схемі консолі. МРЕ надає можливість самостійно вибрати конкретні сенсори системи або налаштувати політикові й надати МРЕ зробити цей вибір. Я поставив МРЕ завдання вибрати системи по швидкості процесора. Потім пішло налаштування політик для Policy Enforce Sensor (шляхом створення іменованої політики через Policy Catalog на деревоподібній схемі консолі), після чого – вибір і призначення її папкам, у яких утримуються системи МРЕ Sensor.

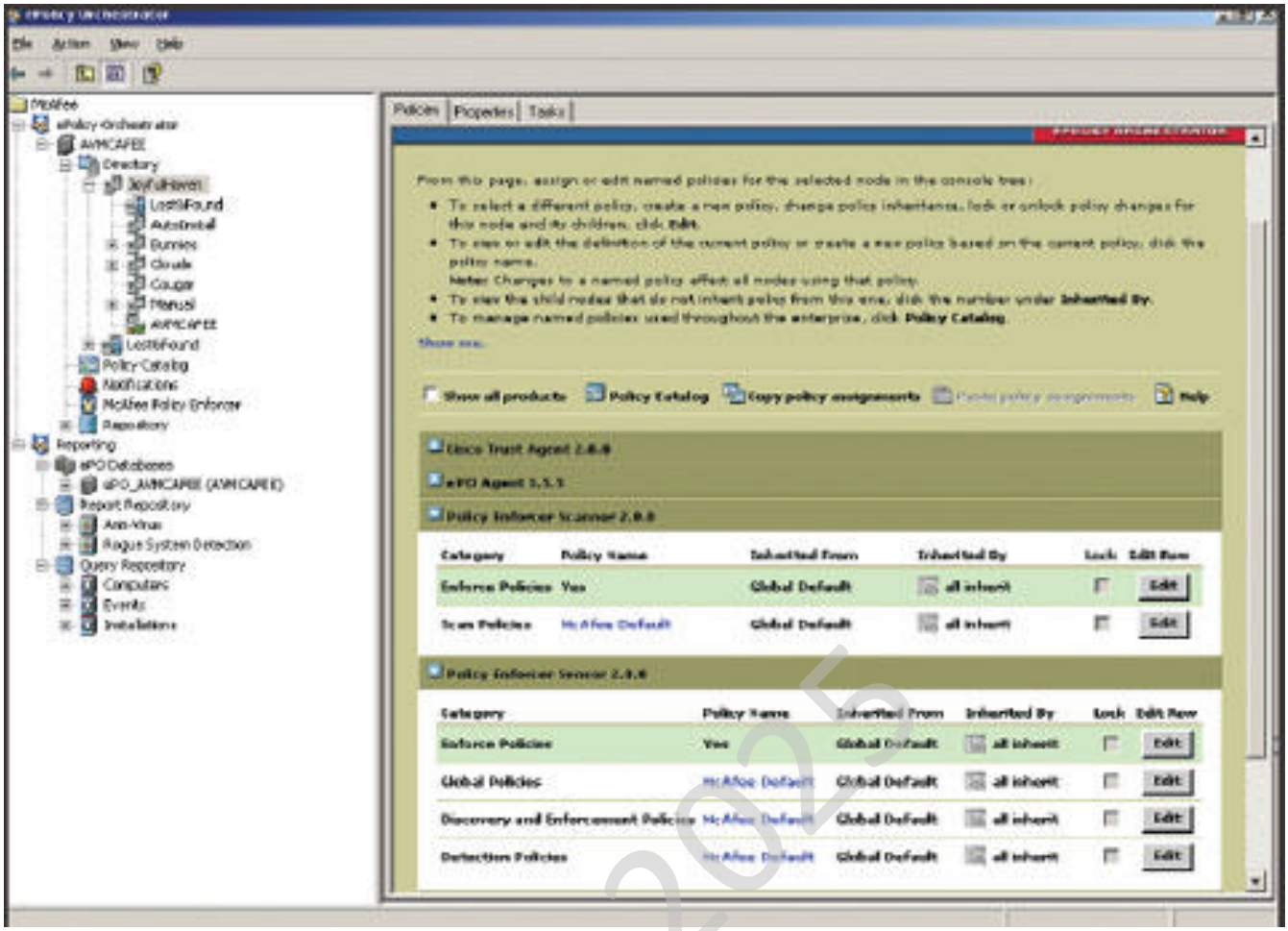

Рисунок 2.3 – Інтерфейс користувача McAfee Policy Enforcer

Для випробування централізованого керування я створив просте правило, що вимагає наявності пакета McAfee AntiVirus Enterprise у системі, і застосував це правило до трьох систем з функціонуючим агентом Enforcer, але без зазначеного антивірусного пакета. По всім трьох системах була видана інформація про їхню невідповідність вимогам безпеки з наступним обмеженням мережного доступу. Я як і раніше одержував відгуки у відповідь на ping-запити інших систем мережі, але не мав доступу ні до загальних мережних ресурсів, ні до віддалених Web-сайтам.

Підсумок. MPE – вдало спроектований і зручний у керуванні продукт. Справжнє задоволення доставляє робота з EPO-консоллю, що забезпечує тісну інтеграцію MPE з іншими елементами інфраструктури забезпечення безпеки

систем і мереж від McAfee. Зокрема, ефективний контроль мережного доступу реалізується з використанням SNMP-керованих комутаторів. Я рекомендую цей варіант тим, хто в цей час може обходитися без 802.1x і DHCP-методів контролю виконання вимог політики безпеки.

StillSecure Safe Access

На відміну від інших учасників даного порівняльного аналізу, Safe Access від StillSecure – це Linux-додаток, установлюваний на «голе залізо». StillSecure надає всім клієнтам підтримку в реалізації продукту; установку продукту для проведення даного дослідження виконав місцевий технічний фахівець.

Архітектура. SafeAccess підтримує перевірку «благонадійності» кінцевих пристроїв без агента, на основі Active, а також у клієнт-агентському режимі. У частині виконання контролю дотримання вимог безпеки підтримується варіант 802.1x і оперативна перевірка до з'єднання, з агентом і на основі DHCP у режимі «після з'єднання». Передбачено також участь в інфраструктурі Cisco NAC.

Адміністратори великих мереж можуть розміщати сервери Safe Access Enforcement – одиничні або групами з балансуванням навантаження – у різних місцях мережі. При такій реалізації всі сервери Enforcement управляються одним сервером.

Інтерфейс керування, що використовує Web-браузер, добре сконструйований і доступний по захищеному з'єднанню HTTPS. Концепція розподіленого адміністрування реалізується із застосуванням чотирьох класів користувальницьких ID – System Administrator, Cluster Administrator, Help Desk і View Only. Як показано на рисунку 2.4, інтерфейс керування передбачає відображення стану всіх виявлених систем мережі разом з контекстною довідковою інформацією.

Як і у випадку з іншими учасниками даного порівняльного аналізу, структура кожної конкретної реалізації Safe Access визначається політиками перевірки й контролю відповідності вимогам безпеки. StillSecure пропонує широкий набір перевірочних тестів, які можна застосовувати до політиків,

включаючи перевірку наявності найпоширеніших додатків забезпечення безпеки, відновлень і налаштувань операційної системи й браузера, а також даних про розповсюджені шкідливі програми. Можна також виконувати перевірку на наявність необхідних або заборонених додатків. Safe Access випускається з різноманітними готовими політиками, що забезпечують високий, середній і низький рівні контролю виконання вимог безпеки. Safe Access автоматично виконує завантаження відновлень для тестів, доступних для використання, але не застосовуваних до діючих політиків автоматично.

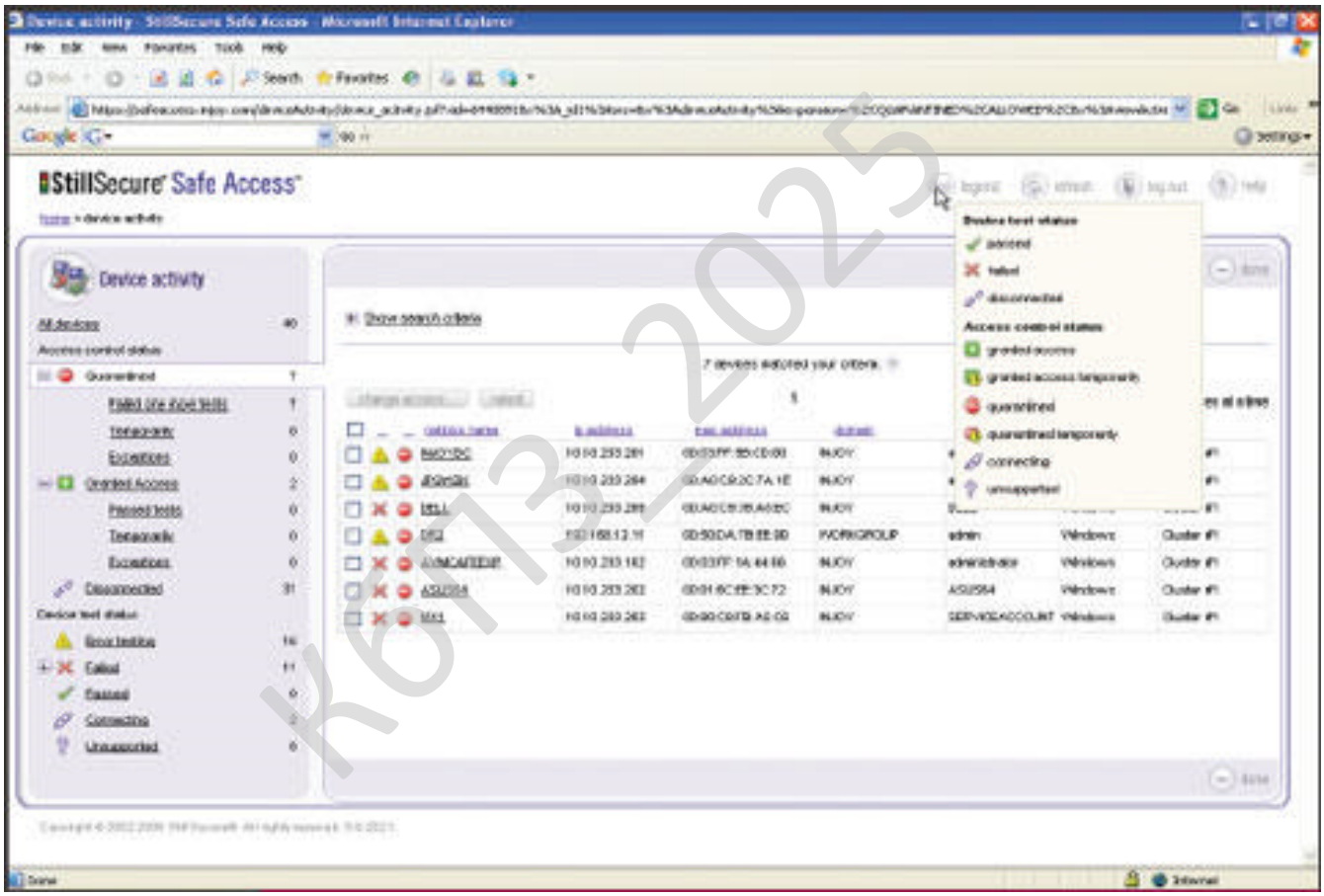

Рисунок 2.4 – Інтерфейс користувача StillSecure Safe Access

Safe Access передбачає безліч функцій, що забезпечують підтримку поетапної, зручної для користувача реалізації NAC, включаючи можливість тимчасового надання мережного доступу системі, що не відповідає вимогам

конкретної політики. Якщо система не проходить тест, можна віддати користувачеві розпорядження вручну внести необхідні виправлення або скористатися що передбачається Safe Access підтримкою деяких популярних додатків автоматизованого виправлення.

Установка. Основна установка, ініціюємe завантаженням сервера з настановного компакт-диску, пройшла швидко. Як і у випадку з іншими продуктами, початкове налаштування конфігурації зайняло набагато більше часу, чим установка самої програми. Для проведення тестування я настроїв Safe Access на контроль виконання вимог безпеки з використанням 802.1x. Налаштування Safe Access на використання ізольованих мереж 802.1x (quarantine networks) вимагає тільки завдання адрес ізольованих підмереж і вибору відмічуваного поля 802.1x. частина, Що Залишилася, налаштування включала установку початкових політик і завдання конфігурації комутатора 802.1x з використанням інформації, що стосується автентифікації й VLAN. Все виконуване після установки побудова конфігурації зайняла менш двох годин.

Після початкової реалізації я досліджував доступні екрани конфігурації й виконав тестування додаткових функцій. Safe Access дозволяє вказати, який із трьох методів тестування – з агентом Safe Access, з агентом Active або без агента – потрібно застосувати, а також порядок реалізації цих методів. Safe Access підтримує три джерела «вірчих даних» для перевірки з метою автентифікації кінцевого пристрою без агента: Windows ID, LDAP і база даних, доступна через інтерфейс Java Database Connectivity (JDBC).

Політики в Safe Access працюють унікальним образом. Кожний сервер Safe Access має один набір політик. Можна виконати спеціалізоване налаштування стандартних політик, а також додати власні політики як в активованому, так і у виключеному стані. Кожній політиці призначається набір доменів Windows або кінцевих пристроїв по діапазоні імен, MAC-адрес, адрес підмережі або IP-адрес, після чого політики розташовуються в логічному порядку. Перевірка кінцевих пристроїв здійснюється відповідно до першої

політики, у відношенні якої ці пристрої задовольняють вимогам приналежності. Кінцеві пристрою, для яких не підходить жодна з політик, перевіряються відповідно до самої останнього – звичайно найменш строгої – політикою зі списку. Для кожного з тестів у рамках політики можна призначити дії, які Safe Access виконує у випадку невдалого результату тесту, що передбачають відправлення електронного повідомлення на адміністративну адресу електронної пошти, негайну або відкладену ізоляцію й звертання до системи автоматизованого виправлення. У випадку невдалого проходження кінцевим пристроєм більше одного тесту, передбаченого політикою, програма призначає найбільш сильне із заданих обмежувальних дій. Я задав як дію повідомлення по електронній пошті й одержав детальний опис причин невдалого результату перевірки кінцевого пристрою. Ця інформація, на мій погляд, є потенційно корисною для довідкової служби Help, що робить користувачам допомога в рішенні проблем. Якщо стандартні тести не відповідають потребам, у посібнику користувача Safe Access можна знайти інформацію про те, як створювати власні тести мовою розробки Python.

Перевірка приміщення в карантин пройшла без несподіванок. Невдалий результат тесту приводив до негайної ізоляції, якщо це передбачало налаштування конфігурації, або до відкладеної ізоляції при відповідній установці в описі тесту. На рисунку 2.4 наведено відображення стану пристроїв. Є можливість негайного надання поміщеному в карантин пристрою додаткового часу, і я зміг виконати повторну перевірку кінцевого пристрою на відповідність вимогам безпеки.

Підсумок. В особі Safe Access мережний адміністратор здобуває чудову комбінацію простоти у використанні, гнучкості в призначенні політик і можливостей забезпечення безпеки мережі. Web-інтерфейс користувача демонструє швидку реакцію, простий для розуміння й насичений корисними контекстними підказками. Хоча для інтерфейсу керування Safe Access не передбачена інтеграція з іншими технологіями забезпечення безпеки (наприклад,

McAfee EPO), це може компенсуватися раціональністю й простотою його конструкції.

На закінчення тестування в мене визначилися два фаворити в даній групі. Safe Access від StillSecure одержує звання «Вибір редакції» за чітку реалізацію 802.1x, гарну керованість і гнучкі можливості приміщення в карантин. Я не перевіряв показники продуктивності, але можна припустити, що Linux-Ядро цього продукту, спеціально орієнтоване на рішення NAC, здатно витримувати більше навантаження. Іншим моїм фаворитом стала технологія Policy Enforcer від McAfee. Мені надзвичайно сподобалася консоль EPO, зокрема вдало спроектована можливість інтеграції засобів керування, якими володіють продукти сімейства технологій безпеки McAfee.

Sophos EndForce Enterprise

Переваги: підтримувані методи реалізації контролю виконання політики безпеки включають 802.1x, DHCP, агентський і VPN; орієнтований на користувача контроль дотримання вимог політики погодиться з характером керування системами в багатьох організаціях.

Недоліки: відносна складність архітектури, що утрудняє керування; відсутність функції виявлення мережних пристроїв.

Рекомендації: працездатна система, але із занадто складною конструкцією, непроста в реалізації й керуванні. Однак для когось орієнтація на користувачів може виявитися достатньою компенсацією цієї незручності.

Dynamic NAC for Windows

Переваги: контроль виконання вимог політики безпеки в режимі ARP-переадресації реалізується з використанням будь-якого мережного комутатора; гнучкі можливості налаштування політик; підтримка агентів Linux і Mac, а також Windows.

Недоліки: дане рішення ставиться до типу post-connect (тобто виконання перевірки після з'єднання), що означає ймовірність помилок процесу контролю; опис політики вимагає уваги до деталей.

Рекомендації: при акуратному налаштуванні конфігурації дане рішення здатне скласти ефективний базовий шар NAC-захисту. Обіцяна підтримка застосування декількох політик і агентів Linux і Mac виявиться вирішальним аргументом для багатьох, однак на даний момент я вважаю, що продукт не готовий до роботи в режимі максимального навантаження.

McAfee Policy Enforcer

Переваги: завдяки керуванню, здійснюваному ePolicy Orchestrator, продукт Policy Enforcer відносно простий у реалізації, структура клієнтських каталогів підтримує автоматичне призначення IP-адрес новим клієнтам; контроль виконання вимог безпеки реалізується з використанням SNMP-керованого комутатора або агента.

Недоліки: відсутність 802.1 x-x- і DHCP-контролю виконання вимог безпеки.

Рекомендації: Policy Enforcer – вдало спроектоване й зручне в керуванні NAC-рішення, особливо для користувачів, що працюють із SNMP-керованими комутаторами з можливістю організації VLAN. Справжнє задоволення доставляє робота з консоллю EPO, а структура прив'язки мережних атрибутів до наборів правил перевірки, а правил – до обмежень мережного доступу надзвичайно зручна.

StillSecure Safe Access

Переваги: широкий діапазон можливостей тестування й контролю виконання політики безпеки, включаючи 802.1x; гнучка й легкореалізована структура політик; модульна структура безпеки консолі, адаптуєма до розподіленого адміністрування; широкі можливості спеціалізованого налаштування тестування кінцевих пристроїв з використанням мови Python, якщо готові тести не відповідають потребам

Недоліки: відсутність підтримки контролю виконання вимог безпеки з використанням SNMP-керованих комутаторів.

Рекомендації: StillSecure створила чудову, просту в налаштуванні й використанні систему NAC. Робота з легкою в керуванні Web-консоллю, стандартними тестами, готовими варіантами контролю виконання вимог безпеки робить приємність. Можливість надавати тимчасовий мережний доступ системам, що невдало пройшли перевірку, напевно зацікавить користувачів.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Як мова програмування обрана Python. Python – високорівнева мова програмування загального призначення з акцентом на продуктивність розроблювача й читаність коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує кілька парадигм програмування, у тому числі структурне, об'єктно-орієнтоване, функціональне, імперативне й аспектно-орієнтоване. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточні обчислень і зручні високорівневі структури даних. Код у Python організовується у функції й класи, які можуть поєднуватися в модулі (які у свою чергу можуть бути об'єднані в пакети).

Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється вільно під дуже ліберальною ліцензією, що дозволяє використовувати його без обмежень у будь-яких застосунках, включаючи пропрієтарні. Є реалізації інтерпретаторів для JVM (з можливістю компіляції), MSIL (з можливістю компіляції), LLVM і інших. Проект PyPy пропонує реалізацію Python на самому Python, що зменшує витрати на зміни мови й постановку експериментів над новими можливостями.

Python – мова програмування, що активно розвивається, нові версії (з додаванням/зміною язикових властивостей) виходять приблизно раз у два з

половиною року. Внаслідок цього й деяких інших причин на Python відсутні ANSI, ISO або інші офіційні стандарти, їхня роль виконує CPython.

Python портований і працює майже на всіх відомих платформах – від КПК до мейнфреймів. Існують порти під Microsoft Windows, практично всі варіанти UNIX (включаючи FreeBSD і Linux), Plan 9, Mac OS і Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 і навіть OS/390, Symbian і Android.

При цьому, на відміну від багатьох портуємих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Більше того, існує спеціальна версія Python для віртуальної машини Java – Jython, що дозволяє інтерпретаторові виконуватися на будь-якій системі, що підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на Python. Також кілька проектів забезпечують інтеграцію із платформою Microsoft .NET, основні з яких – IronPython і Python.Net.

Python підтримує динамічну типізацію, тобто тип змінної визначається тільки під час виконання. Тому замість «присвоювання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». У Python є убудовані типи: бульові, рядки, Unicode-рядки, цілі числа довільної точності, числа із плаваючої коми, комплексні числа й деякі інші. З колекцій Python підтримує кортежі (*tuples*), списки, словники (асоціативні масиви) і, починаючи з версії 2.4, безлічі. Всі значення в Python є об'єктами, у тому числі функції, методи, модулі, класи.

Додати новий тип можна або написавши клас (class), або визначивши новий тип у модулі розширення (наприклад, написаному мовою C). Система класів підтримує спадкування (одиночне й множинне) і метапрограмування. Можливе спадкування від більшості убудованих типів і типів розширень.

Всі об'єкти діляться на посилальні й атомарні. До атомарного ставляться int, long, complex і деякі інші. При присвоюванні атомарних об'єктів копіюється їхнє значення, у той час як для посилальних копіюється тільки покажчик на об'єкт, таким чином, обидві змінні після присвоювання використовують те саме

значення. Посилальні об'єкти бувають змінювані й незмінні. Наприклад, рядки й кортежі є незмінними, а списки, словники й багато інших об'єктів – змінюваними. Кортеж у Python є, по суті, незмінним списком. У багатьох випадках кортежі працюють швидше списків, тому якщо ви не плануєте змінювати послідовність, то краще використовувати саме їх.

Мова має чіткий і послідовний синтаксис, продуману модульність й масштабованість, завдяки чому вихідний код написаних на Python програм легко читаємий.

Python – стабільна й розповсюджена мова. Він використовується в багатьох проектах і в різних якостях: як основна мова програмування або для створення розширень і інтеграції застосунків. На Python реалізоване велика кількість проектів, також він активно використовується для створення прототипів майбутніх програм. Python використовується в багатьох великих компаніях.

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 доступу у мережу на основі технології Cisco NAC.

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

Переваги NAC

Хоча розробка технологій забезпечення безпеки ведеться не перший рік, і на їхню реалізацію витрачені мільйони доларів, віруси, хробаки, шпигунські й інші злочинні програми за даними Звіту про безпеку CSI/FBI за 2014 рік залишаються на сьогодні основною проблемою, з якою зіштовхуються компанії. Значна кількість виникаючих інцидентів приводить до істотних фінансових збитків через змушені простої, недоотриманих прибутків, ушкодження або знищення інформації й зниження продуктивності.

Висновок очевидний: для рішення цієї проблеми недостатньо традиційних рішень по забезпеченню безпеки. Cisco Systems® розробила комплексне рішення по забезпеченню безпеки, у якому об'єднані кращі інструменти боротьби з вірусами, безпеки й керування. Це гарантує дотримання політики безпеки всіма пристроями, що перебувають у мережному середовищі. NAC дає вам можливість проаналізувати й проконтролювати всі пристрої, що звертаються до вашої мережі. Стежачи за тим, щоб на кожному кінцевому пристрої дотримувалася корпоративна політика безпеки (наприклад, за тим, щоб на цих пристроях діяли оновлені й найбільш підходящі ресурси забезпечення безпеки), компанії можуть істотно скоротити або взагалі звести до нуля кількість кінцевих пристроїв, які стають традиційним джерелом зараження або злому мереж.

Істотне підвищення рівня безпеки

Хоча в більшості компаній для автентифікації користувачів застосовується керування ідентифікацією, авторизації й аудита (AAA) і авторизації мережних привілеїв, фактично відсутній спосіб автентифікації профілю безпеки кінцевого пристрою, на якому працює користувач. Без точного способу оцінки “стану

здоров'я" пристрою навіть самий надійний користувач може ненавмисно піддати інших користувачів мережі значному ризику, що виникає через присутність інфікованого або недостатньо захищеного від інфекцій пристрою.

NAC – це набір технологій і рішень, фундаментом яких служить загальногалузева ініціатива, реалізована під патронажем Cisco Systems.

NAC використовує інфраструктуру мережі для контролю над дотриманням політики безпеки на всіх пристроях, що прагнуть одержати доступ до ресурсів мережі.

Цим шляхом знижується збиток, що можуть заподіяти виникаючі погрози безпеки. Використовуючи NAC, клієнти одержують можливість надавати мережний доступ тільки дотримує запропоновані вимоги, безпечним кінцевим пристроям (наприклад, комп'ютерам, серверам і КПК) і обмежувати доступ для пристроїв, не відповідним вимогам.

NAC – унікальна технологія, оскільки вона призначена для інтеграції в інфраструктуру мережі. Чому контроль над дотриманням правил політик і стратегію верифікації варто реалізовувати саме в мережі?

Практично кожний біт інформації, що представляє для компанії інтерес або цінність, прямо або побічно є присутнім у мережі.

Практично кожний пристрій, що представляє для компанії інтерес або цінність, з'єднано із цією же мережею.

Впровадження контролю доступом у мережу дає компанії можливість розгорнути найбільш повномасштабне рішення по забезпеченню безпеки, що охоплює максимальна кількість пов'язаних з мережею пристроїв.

Ця стратегія використовує існуючу інфраструктуру й ресурси компанії в області безпеки й керування, тому вона накладає на ресурси IT мінімальне додаткове навантаження.

Після впровадження NAC при кожній спробі кінцевого пристрою з'єднатися з мережею пристрій мережного доступу (LAN, WAN, бездротового або віддаленого доступу) автоматично запитує профіль безпеки кінцевого пристрою,

що видається за допомогою інстальованого клієнта або інструментів експертної оцінки. Потім ця профільна інформація зіставляється з мережною політикою безпеки, і рівень відповідності пристрою цим політикам визначає реакцію мережі на запит доступу. Мережа може просто дозволяти доступ або відмовляти в ньому або ж обмежувати доступ, переадресуючи пристрій у сегмент мережі, у якому обмежений контакт із потенційно уразливими вузлами. Не відповідним вимогам пристрій також можна помістити в карантин шляхом переадресації на коректувальний сервер, де в нього будуть внесені відновлення, які забезпечать дотримання політик.

Зокрема, NAC може виконувати наступні перевірки дотримання політики безпеки:

– Чи працює на пристрої авторизована версія операційної системи.

– Чи внесені в ОС належні програмні виправлення або чи отримані самі останні відновлення.

– Чи інстальовано на пристрої антивірусне програмне забезпечення й чи є новітній набір файлів сигнатур.

– Чи активовані антивірусні ресурси й чи запускалися вони в недавній час.

– Чи інстальовані й чи правильно конфігурований персональний міжмережний екран, інструменти запобігання вторгнень або інше програмне забезпечення безпеки ПК.

– Чи вносилися модифікації або несанкціоновані зміни в корпоративний образ пристрою.

Відповіді на ці й аналогічні питання, пов'язані із профілем безпеки, потім використовуються для винесення логічно-обґрунтованих, заснованих на політику рішень по доступі в мережу.

Впровадження NAC забезпечує, зокрема, що випливають переваги:

– Істотно зростає рівень безпеки в будь-якій мережі, незалежно від її розміру або складності структури. NAC допомагає простежити за дотриманням політики безпеки на всіх користувальницьких мережних пристроях. За рахунок

превентивного захисту від хробаків, вірусів, шпигунських і інших злочинних програм компанії одержують можливість зосередити зусилля на профілактиці, а не на відповідних мірах.

– Завдяки широкому визнанню, застосуванню й інтеграції із продуктами провідних розроблювачів більш ефективно освоюються існуючі інвестиції в мережну інфраструктуру Cisco, а також у програмні засоби боротьби з вірусами, безпеки й керування.

– Зростає стабільність роботи компаній і масштабованість, оскільки є можливість інспектування й контролю всіх пристроїв, що з'єднуються з мережею, незалежно від того, який метод доступу вони використовують (зокрема, маршрутизатори, комутатори, бездротової, що комутирується доступ, VPN).

– Не відповідають запропонованим вимогам і некеровані кінцеві пристрої не можуть вплинути на доступність мережі й продуктивність роботи користувачів.

– Скорочуються експлуатаційні витрати, пов'язані з ідентифікацією й санацією не відповідають вимогам, некерованих і заражених систем.

Варіанти впровадження NAC

Cisco пропонує підходи до впровадження NAC на базі пристроїв і на базі архітектури, які відповідають функціональним і операційним запитам будь-якої компанії. При цьому компаніям може бути необхідна як найпростіша політика безпеки, так і підтримка комплексних структур безпеки, у яких інструменти забезпечення безпеки різних розроблювачів об'єднані з корпоративним рішенням по керуванню настільними пристроями.

Пристрій NAC (NAC Appliance), реалізоване в лінійці продуктів Cisco Clean Access, забезпечує швидке розгортання із сервісами автономної експертизи кінцевих вузлів, керування політикою й санації. На додаток до цього, Архітектура NAC (NAC Framework) поєднує інтелектуальну мережну інфраструктуру з рішеннями більше 50 розроблювачів провідних програмних засобів боротьби з вірусами, безпеки й керування.

Пристрій контролю доступу в мережу (NAC Appliance)

Продукти NAC Appliance, що входять у лінійку продуктів Cisco Clean Access, забезпечують швидке розгортання із сервісами автономної експертизи кінцевих вузлів, керування політикою й санації. Ця швидко впроваджуване “коробкове рішення” ("in-a-box"), автоматично виявляє, ізолює й очищає інфіковані або уразливі провідні або бездротові кінцеві точки, що намагаються одержати доступ у мережу.

Cisco Clean Access виконує три найважливіші функції захисту:

– Розпізнає користувачів, їхнього пристрою, їхня роль у мережі й точку авторизації, автентифікації.

– Оцінює статус безпеки кінцевих вузлів, використовуючи технологію сканування й аналізу або легку програму-агент для більше глибокої експертизи статусу й перевірки уразливостей.

– Реалізує політикові безпеки в мережі, блокуючи, поміщаючи на карантин і проводячи санацію кінцевих вузлів, що не відповідають вимогам політик.

Cisco Clean Access так само надає наступні переваги при розгортанні:

– Масштабованість – Cisco Clean Access можна розгорнути негайно для рішення існуючих проблем входу в мережу, а потім продовжити доробку й оцінку архітектури NAC, оскільки елементи Cisco Clean Access можна інтегрувати в більше широку архітектуру NAC.

– Швидке розгортання – Cisco Clean Access – це компактне, “готове до вживання” рішення, у якому передбачена підтримка відновлень ресурсів боротьби з вірусами, шпигунськими програмами й відновлень Microsoft.

– Гнучкість – Cisco Clean Access підтримує різнорідну мережну інфраструктуру з різними настільними операційними системами.

Характеристики мережі, ідеальні для впровадження Cisco Clean Access:

– Мережа LAN не підтримуючий протокол 802.1x.

– Бездротові, філіальні, віддалені або прості робочі LAN.

– Централізоване робітниче середовище й керування IT.

– Доступ до мережі некерованих комп'ютерів (наприклад, відвідувачів, підрядників або учнів).

– Неоднорідна (яка включає продукти багатьох вендорів) мережна інфраструктура.

Архітектура контролю доступу в мережу (NAC Framework Solution)

NAC так само пропонується як архітектура, що додатково зміцнює існуючі основи, складені з мережних технологій Cisco і впроваджені рішення інших розроблювачів в області керування й забезпечення безпеки. У рішенні NAC Framework закладені наступні переваги:

– Повномасштабний контроль за рахунок оцінки всіх кінцевих вузлів і охвату всіх методів доступу, включаючи LAN, бездротовий, віддалений доступ і WAN.

– Видимість кінцевих вузлів і контроль над ними для гарантії дотримання корпоративної політики безпеки на керованих, некерованих, гостьових і ушкоджених пристроях.

– Протягом усього життєвого циклу забезпечена підтримка ресурсів контролю на кінцевих вузлах, які автоматизують оцінку, автентифікацію, авторизацію кінцевих вузлів і прийняття корективних мір.

– Об'єднання централізованого керування політикою, інтелектуальних мережних пристроїв і мережних сервісів з рішеннями десятків розроблювачів в області боротьби з вірусами, забезпечення безпеки й керування для деталізованого контролю доступу в мережу.

– Підтримка багатої “екосистеми” партнерів і технологій за рахунок, стандартів і гнучких API, які дають стороннім розроблювачам можливість внести свою лепту в загальне рішення.

Наступні характеристики мережі оптимальні для впровадження архітектури NAC:

– Великомасштабні корпоративні структури.

– Комплексне робітниче середовище з LAN/WAN/бездротовими ресурсами.

– Інфраструктура LAN/WAN/бездротових ресурсів, повністю або в основному заснована на технологіях Cisco.

– Операційна сумісність із рішеннями партнерів по NAC в області забезпечення безпеки й керування.

– Впровадження ресурсів IP-Телефонії або плановані впровадження.

– Впровадження ресурсів стандарту 802.1X або плановані впровадження.

Захист інвестицій

Cisco пропонує найбільш повний набір продуктів і рішень по контролю доступу в мережу, здатний задовольнити функціональні потреби будь-якої компанії. Оскільки потреби багатьох компаній згодом міняються, складові частини продукту Cisco Clean Access, які інстальовані зараз, можна буде використовувати для підтримки наступного впровадження архітектури NAC.

Незалежно від того, який підхід, на думку, підходить для вашого робітничого середовища, технології Cisco NAC служать для захисту інвестицій у відповідну мережну технологію. Одночасно із цим, можливість спільної роботи й функціональна сумісність допомагають забезпечити плавний перехід від Cisco Clean Access до архітектури NAC, що пропонує додаткові можливості й переваги.

3.2 Розробка структурної схеми

Планування, розробка й впровадження ефективного рішення по NAC

Для того щоб розгорнуте рішення Cisco NAC було ефективним, відділ обслуговування Cisco Advanced Services визначає наступні послуги з аналізу потреб, планування, розробки й впровадження:

– Оцінка готовності до NAC – Аналіз вимог, пропонованих до розгортання, і експертиза готовності мережних пристроїв, процесів і архітектури до підтримки NAC.

– Обмежене розгортання NAC – Інсталяція й конфігурування обмеженого рішення, що дає фахівцям можливість випробувати NAC і одержати досвід роботи з ним.

– Прорблення NAC – Допомога фахівцям у детальному пророблені для інтеграції NAC у мережну інфраструктуру.

– Інженерна підтримка впровадження NAC – Допомога фахівцям у повномасштабному впровадженні: розробка докладних планів по інсталяції, конфігуруванню, інтеграції й керуванню й виконання інсталяції, конфігурування й тестування на об'єкті. Це забезпечить плавну інтеграцію впроваджуваних ресурсів у ваше виробниче середовище.

Після цього виконуються наступні дії:

– Розгорніть Cisco Clean Access зараз. Cisco Clean Access дозволяє вам негайно впровадити й ефективно використовувати рішення по контролі доступу в мережу.

– Визначитеся, чи буде потрібна архітектура NAC. Впровадивши Cisco Clean Access, можете оцінити й те, чи відповідаєте профілю підходу на базі архітектури. Вибираючи для впровадження будь-яке рішення NAC, потрібно враховувати ряд факторів, у тому числі й особливості мережі, у якій впроваджується рішення, і компанії, що його впроваджує.

– Ефективно використовуйте інвестиції в Cisco Clean Access. Елементи Cisco Clean Access можна повністю інтегрувати в архітектуру NAC.

Технологія NAC

Компоненти пристрою NAC (NAC Appliance)

Cisco Clean Access включає наступні елементи:

– Cisco Clean Access Server виконує експертизу пристроїв і вводить привілеї доступу, засновані на дотриманні запропонованих вимог на кінцевих вузлах.

– Cisco Clean Access Manager забезпечує централізоване керування рішенням Cisco Clean Access, включаючи сервіси контролю дотримання правил політик і прийняття коррективних мір.

– Cisco Clean Access Agent – опціональна безкоштовна клієнтська програма, що забезпечує більше тверду експертизу дотримання правил політик у кінцевих вузлах і оптимізовані корректівні міри в керованих і в некерованих робітничих середовищах.

В Cisco Clean Access забезпечена підтримка бездротового доступу із застосуванням наступних технологій:

– Всі точки доступу 802.11 Wi-Fi, включаючи точки доступу Cisco Aironet.

– Будь-які клієнтські пристрої Wi-Fi із суплікантом IEEE 802.1X, що підтримує NAC.

Компоненти архітектури NAC (NAC Framework)

NAC Framework забезпечує наступну технологічну підтримку:

– Широка підтримка мережних пристроїв для LAN колективного користування, WAN, VPN і точок бездротового доступу.

– Можливість з'єднання зі сторонніми інструментами експертизи хостов для роботи із пристроями, що діють без оператора, без програм-агентів і з іншими не реагуючими пристроями (nonresponsive devices), і можливість застосування різних правил політики до кожного конкретного пристрою.

– Широка підтримка платформ для Cisco Trust Agent.

– Розширення інтеграції продуктів різних розроблювачів з перевірками статусу додатків і операційних систем, які виходять далеко за рамки антивірусних відновлень і базових програмних виправлень для операційних систем.

Архітектура NAC підтримується наступними технологіями:

– Маршрутизатори Cisco: маршрутизатори з інтегрованими сервісами серій Cisco 83x, 18xx, 28xx і 38xx; модульні маршрутизатори доступу 1701, 1711, 1712, 1721, 1751, 1751-V і 1760; мульти-сервісні маршрутизатори доступу 2600XM, 2691, 3640 і 3660-ENT; маршрутизатори серії 72xx.

– Комутатори Cisco: Cisco Catalyst 6500 Series Supervisor Engine 2, 32, і 720, з Cisco Catalyst OS, Cisco IOS® Software, або гібридні варіанти (підтримка

Cisco IOS Software на Supervisor Engine 32 і 720), Cisco Catalyst 4000 Series Supervisor Engine II+, II+TS, IV, V, і V-10GE, з Cisco IOS Software, Cisco Catalyst 4948 і 4948-10GE, Cisco Catalyst 3550, 3560, 3750 с Cisco IOS с IP-Base and IP-Services, Cisco Catalyst 2940, 2950, 2955, 2960, 2970

– Бездротової доступ Cisco: точки доступу Cisco Aironet, спрощені точки доступу Cisco Aironet, з'єднані з контролером бездротових LAN Cisco, сервісний модуль бездротової LAN (WLSM) Cisco Catalyst серії 6500 і всі пристрої Cisco Aironet, Cisco Compatible, і клієнтські Wi-Fi пристрою із суплікантом IEEE 802.1X, що підтримує NAC.

– Концентратори Cisco VPN серії 3000.

– Програма-агент Cisco Trust Agent.

– Сервер контролю безпечного доступу Cisco ACS.

– Програмне забезпечення сторонніх розроблювачів.

Елементи, що рекомендуються:

– Програма-агент безпеки Cisco Security Agent.

– Система моніторингу, аналізу й відповідних заходів щодо забезпечення безпеки Cisco Security Monitoring, Analysis and Response System (MARS).

– Рішення Cisco по безпечному керуванню інформацією CiscoWorks Security and Information Management Solution (SIMS).

По своїй суті NAC Framework – це архітектурно-архітектурно-орієнтована технологічна концепція, в основі якої лежить галузева ініціатива компанії Cisco Systems. Ця концепція передбачає використання мережної інфраструктури й ПЗ сторонніх виробників для контролю за дотриманням політик безпеки для всіх кінцевих пристроїв. Крім того, концепція дозволяє інтегрувати мережні технології Cisco, антивірусні пакети й інші програмні продукти для захисту мережі й керування мережею, а також підвищити віддачу від інвестицій у названі компоненти інфраструктури.

Система NAC Framework дозволяє застосовувати привілею доступу на мережних пристроях Cisco, включаючи маршрутизатори й комутатори, при

спробі кінцевого пристрою підключитися до керованої мережі. Рішення про надання доступу до мережі приймається виходячи з відомостей про кінцевий пристрій, наприклад, про поточний стан його програмного забезпечення, зокрема, про наявність антивірусних програм і рівень патчів для операційної системи. Розробка NAC Framework дозволяє вибрати один із трьох варіантів дій відносно пристроїв, що не пройшли перевірку на відповідність вимогам безпеки: їм можна заборонити доступ, їх можна помістити в карантинну зону й, нарешті, їм можна надати обмежені права на доступ до мережних ресурсів.

Рішення NAC Framework перевіряє відповідність пристроїв політиці безпеки мережі WLAN на точках доступу в момент, коли клієнти WLAN намагаються підключитися до мережі. Точки доступу WLAN реалізують політикові NAC за допомогою мереж VLAN. Сервер RADIUS поміщає не відповідають вимогам політики безпеки клієнтів WLAN у карантинну або "лікувальну" мережу VLAN/мобільну групу на автономній точці доступу (рисунок 3.1) або на контролері бездротової мережі у випадку "полегшених" точок доступу.

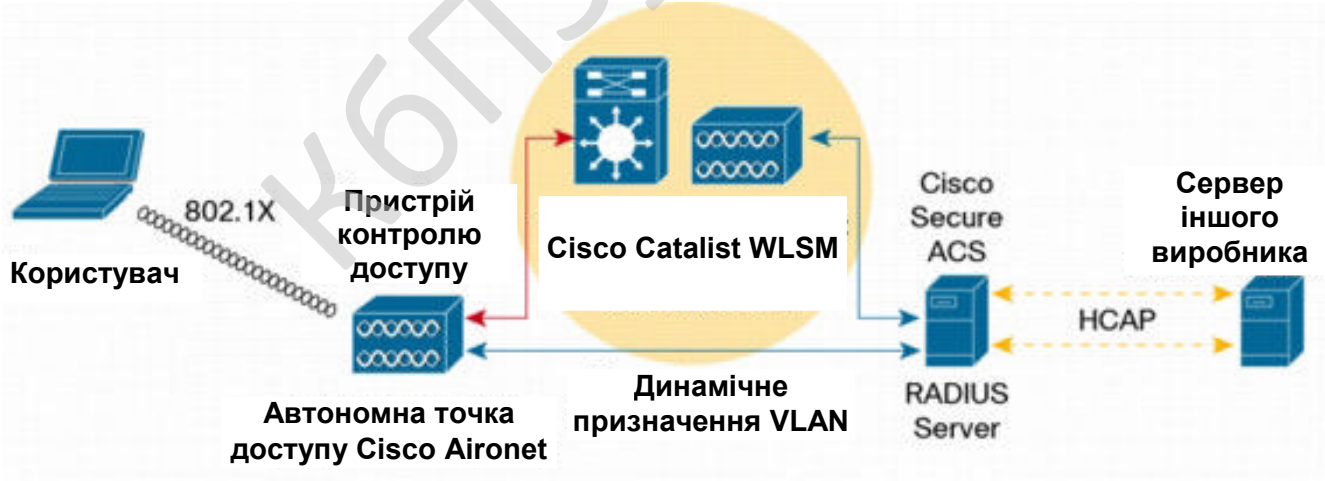

Рисунок 3.1 – Структурна схема системи

NAC Framework, крім маршрутизаторів і комутаторів Cisco, підтримує наступні мережні продукти:

1. Автономні точки доступу Cisco Aironet, які функціонують в ізольованому режимі або в режимі WDS. Сюди входять точки доступу Cisco Aironet серій 1100, 1130AG, 1200, 1230AG, 1240AG і 1300, що працюють під керуванням Cisco IOS Software Release 12.3(7)JA або більше пізньої версії.

2. "Полегшені" точки доступу Cisco Aironet, використовувані в сполученні з контролером бездротової мережі Cisco (точки доступу Cisco Aironet серій 1000, 1130AG, 1200, 1230AG, 1240AG і 1500, контролери бездротової мережі Cisco серій 2000, 4100 або 4400, а також модуль бездротових сервісів (WiSM) для Cisco Catalyst серії 6500 і модуль контролера бездротової мережі для маршрутизаторів з інтеграцією сервісів), що працюють під керуванням Cisco Unified Wireless Network Software Release 3.1 або більше пізньої версії.

3. Модуль бездротових сервісів (WLSM) для Cisco Catalyst серії 6500, використовуваний як пристрій WDS, що працює під керуванням Cisco IOS Software Release 1.4.1 або більше пізньої версії.

4. Будь-які клієнтські пристрої з підтримкою стандарту 802.11 Wi-Fi з доповненням IEEE 802.1X, що підтримує NAC:

– Клієнтський пристрій Cisco Aironet з доповненням NAC від сторонніх виробників, наприклад, від клієнта Funk Odyssey або Meeting House AEGIS.

– Клієнтський пристрій з підтримкою Wi-Fi з доповненням NAC від сторонніх виробників, наприклад, від клієнта Funk Odyssey або Meeting House AEGIS.

– Сумісні з Cisco клієнтські пристрої з версією програмного забезпечення Cisco Compatible 4.0 і вище (версія 4.0 Cisco Compatible включає необхідне доповнення NAC.)

3.3 Розробка функціональної схеми

Для подолання труднощів у слабоформалізованих ситуаціях більше високий якісний рівень оперативного управління припускає забезпечення необхідної й достатньої інтелектуальної підтримки. Запропонована в роботі функціональна схема системи інтелектуальної підтримки (СІП) оперативного управління контролю доступу у мережу на основі технології Cisco NAC наведена на рисунку 3.2.

У системі інтелектуальної підтримки оперативного управління контролю доступу у мережу на основі технології Cisco NAC пропонується використовувати інтелектуальні технології:

– механізм нечіткого логічного виводу для чисельної оцінки ймовірності атаки;

– організоване впорядкування інформації про події в базі знань;

– моделі протидії погрозам;

– прийняття рішень на вибір раціонального варіанта реагування на події безпеки.

Через необхідність максимальної структуризації розроблювальної системи й рішень, пропонується трьохрубіжна модель захисту, що щонайкраще задовольняє всієї сукупності умов її розробки, експлуатації й удосконалення. Трьохрубіжна модель захисту – неформалізований опис комплексу програмно-апаратних засобів захисту, що є основою для розробки системи захисту:

– перший рубіж – периметр об'єкта захисту – набір функціональних підсистем, що включають засоби захисту від зовнішніх вторгнень зловмисника й потенційно можливих погроз віддаленого користувача;

– другий рубіж – набір засобів захисту мережного сегмента від віддалених і локальних мережних вторгнень;

– третій рубіж містить у собі набір засобів захисту окремого персонального комп'ютера або сервера.

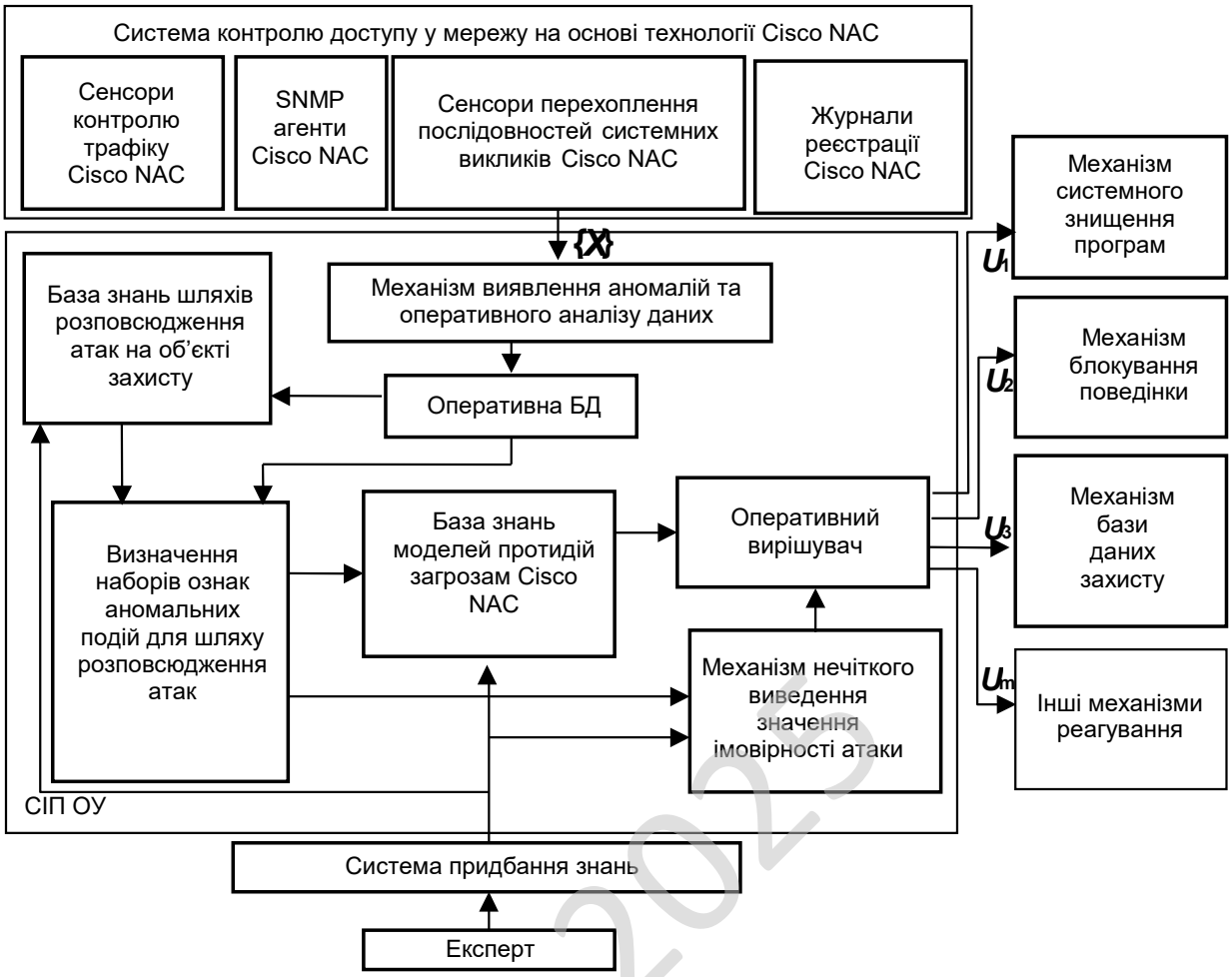

Рисунок 3.2 – Функціональна схема системи

У процесі організаційно-технічного управління, планування ЗІ як функція управління контролю доступу у мережу на основі технології Cisco NAC являє собою процес послідовного зняття невизначеності щодо структури й состава засобів захисту в СЗІ. Процес планування $Р_{пл}$ раціональних наборів ЗЗ характеризується за допомогою вираження:

$$P_{\text{пл}} = \Phi \to S_r,$$

де Ф – множина функціональних підсистем для рубежу захисту;

S_r – обраний набір засобів захисту.

На першому етапі задається множина функціональних підсистем для рубежів захисту, результатом планування є командна інформація, що містить

конкретні дані по розподіляються ресурсам, що, що направляється на досягнення цільового стану СЗІ.

Процес ухвалення рішення про вибір раціонального варіанта набору ЗЗ для рубежу захисту – це функція перетворення змісту інформації про вимоги, запропонованих до засобів захисту, що входить у набір, про характеристики засобів захисту, у підмножину найкращих варіантів набору $S' \subseteq S$. Множина варіантів набору:

$$S = \{S_1, \ldots, S_r, \ldots, S_R\},$$

де R – число варіантів альтернативних наборів, з яких здійснюється вибір.

Для вибору раціонального варіанта набору засобів захисту використовується цільова функція J:

$$S_r = J(S).$$

Сукупність відомостей, що дозволяють зіставляти варіанти наборів, це характеристики засобів захисту функціональних підсистем для рубежу – множина W, що включає в себе дві підмножини:

$$W_{\text{зщ}_l} \subset W_l \text{ і } W_{\text{и}_l} \subset W_l,$$

де $W_{\text{зщ}_l}$ – показник засобів захисту «захищеність інформації»;

$W_{\text{и}_l}$ – показник засобів захисту «витрати» для l-ої функціональної підсистеми.

На основі морфологічного підходу модель прийняття рішень на вибір раціонального варіанта набору може бути представлена у вигляді кортежу:

$$\text{ПР: } \langle \text{Ц}, \Phi, \Pi_s, S, W_l, J, S_r(S') \rangle,$$

де Ц – ціль ухвалення рішення;

Ф – вихідні дані для породження варіантів набору засобів захисту:

$$\Phi = \{\Phi_1, \Phi_2, \ldots, \Phi_l, \ldots, \Phi_L\};$$

Π_s – правило породження варіантів набору, що може бути представлене в аналітичному виді як векторний добуток множин:

$$S = \Phi_1 \times \Phi_2 \times \ldots \times \Phi_l \times \ldots \times \Phi_L,$$

де Φ_l – множина, що складається із засобів захисту l-ої функціональної підсистеми:

$$\Phi_l = \{A_{l1}, A_{l2}, \ldots A_{lm} \ldots, A_{lK_l}\};$$

S – множина породжених варіантів набору;

W_l – дані для вибору раціональних варіантів;

J – цільова функція для вибору раціонального набору засобів захисту(правило вибору);

S_r – раціональний набір засобів захисту.

Відзначається, що в умовах автоматизованого управління й при використанні експертної інформації в процесі ухвалення рішення можна говорити (навіть у випадку формалізованого правила вибору) про раціональне, а не оптимальне рішенні.

Відповідно до трьохрубіжної моделі захисту, основою планування раціонального модульного состава СЗІ є функціональні вимоги до наборів ЗЗ для кожного рубежу, які формулюються на основі нормативної документації, відповідно до рівня критичності оброблюваної інформації. Альтернативні засоби захисту для кожної функціональної підсистеми набору засобів захисту вибираються з урахуванням цих вимог. Варіантів наборів, сертифікованих по необхідному класі захищеності, може бути багато. Порівняння варіантів наборів засобів захисту пропонується робити по кількісній мері.

Для рішення завдання вибору раціональних варіантів наборів засобів захисту для рубежів захисту розробляється метод обробки знань, що використовує неформалізуємий досвід експерта в області ЗІ, що забезпечує перетворення відомостей про характеристики засобів захисту з бази знань і вивід рішення в аналітичній формі – метод формування раціонального комплексу засобів захисту для СЗІ.

1. Розробляються варіанти набору ЗЗ. Множина можливих варіантів рішення завдання вибору задається морфологічною матрицею. Розробляються морфологічні матриці засобів захисту для трьох рубежів.

2. Заповнюються допоміжні матриці, у яких відзначаються сумісні один з одним програмно-апаратні засоби. Допоміжна квадратна матриця сумісних рішень заповнюється в такий спосіб: для кожної пари засобів захисту різних функціональних підсистем визначається, чи сумісні вони, і результат заноситься в таблицю. Якщо ЗЗ сумісні, то функція сумісності $s(A_{lm}, A_{pr}) = 1$, у противному випадку $s(A_{lm}, A_{pr}) = 0$.

3. Генерується безліч рішень на вибір варіантів набору ЗЗ із усіканням цієї множини до підмножини варіантів набору із сумісних між собою програмно-апаратних продуктів.

Множина $S = \{S_1, ...S_r, ..., S_R\}$, що складається із всіх можливих варіантів побудови набору ЗЗ для рубежу, є декартовим добутком множин альтернатив (рядків морфологічної матриці).

Елемент множини:

$$S_r = \left\{ \left(A_{1i}, A_{2j}, \dots A_{lm}, \dots A_{Ln} \right) : A_{lm} \in \Phi_l, \forall l = \overline{1, L} \right\},$$

де L – число функціональних підсистем для рубежу;

A_{lm} – засіб захисту для реалізації l-ой функціональної підсистеми.

Генерація множин рішень на вибір варіантів набору, що складаються із сумісних між собою ЗЗ, здійснюється в такий спосіб.

Відбувається ітераційний синтез варіантів набору, що складаються із сумісних ЗЗ: на першому кроці перебираються послідовно варіанти засобів захисту для першої підсистеми, після вибору альтернативи A_{1i} здійснюється перехід до другого кроку. На другому кроці виконується послідовний перебір варіантів засобів захисту другої підсистеми, але вибір здійснюється тільки для таких альтернатив A_{2j}, для яких функція сумісності $s(A_{1i}, A_{2j}) = 1$ і т.д. При виборі альтернатив з l-ої підсистеми вибір здійснюється тільки з таких альтернатив A_{lm}, для яких функції сумісності дорівнюють одиниці: $s(A_{l-1,m}, A_{lm}) = 1$, $s(A_{2j}, A_{lm}) = 1$, $s(A_{1i}, A_{lm}) = 1$... Таким, образом, вибір ЗЗ із

кожного рядка морфологічної матриці (по одній з кожного рядка) для формування варіанта набору здійснюється тільки із сумісних між собою програмно-апаратних продуктів.

4. Подальше усікання множини S виконується методом повного перебору по заданій цільовій функції. Як цільова функція для вибору варіанта набору $S_r = \{A_{1i}, A_{2j}, \ldots A_{lm}, \ldots A_{Ln}\}$, застосовується функція:

$$J = \max_r \frac{W_{K^1_{\text{зщ}}}^{A_{1i}} + \ldots + W_{K^1_{\text{зщ}}}^{A_{lm}} + \ldots + W_{K^1_{\text{зщ}}}^{A_{Ln}}}{W_{K^1_{\text{и}}}^{A_{1i}} + \ldots + W_{K^1_{\text{и}}}^{A_{lm}} + \ldots + W_{K^1_{\text{и}}}^{A_{Ln}}},$$

де $W_{K^1_{\text{зщ}}}^{A_{lm}}$ – значення показника «захищеність»;

$W_{K^1_{\text{и}}}^{A_{lm}}$ – значення показника «витрати» засобу захисту A_{lm}.

Для оцінки засобів захисту різних функціональних підсистем наборів розробляються ієрархічні структури узагальнених критеріїв якості засобів захисту: показник «захищеність» і показник «витрати».

Критерії якості засобів захисту по ієрархії «захищеність» діляться на дві групи: показники забезпечення ефективності оперативних методів захисту й показники функціональної придатності. Критерії якості по ієрархії «витрати» діляться також на дві групи: у першу включена вартість відповідного засобу захисту, число користувачів по однієї ліцензії й інші можливі економічні витрати; до другої групи витрат ставляться функціональні витрати, такі, наприклад, як падіння продуктивності інформаційної системи при використанні даного засобу захисту.

Оцінка засобів захисту й критеріїв здійснюється попарним порівнянням по методу Т. Сааті, результати приводяться в числовому виді. З використанням ієрархічних структур критеріїв якості ЗЗ обчислюються нормовані значення власних векторів засобів захисту за всіма критеріями до показників «захищеність» $K^1_{\text{зщ}}$ і «витрати» $K^1_{\text{и}}$ на підставі обробки всіх матриць попарних порівнянь із урахуванням зв'язків критеріїв.

Після вибору раціональних наборів засобів захисту для рубежів захисту отриманий раціональний модульний состав цілісного комплексу засобів захисту об'єкта, що задовольняє вимозі

$$J \rightarrow \max.$$

5. Оцінюється, чи задовольняє сформований комплекс засобів захисту вимозі:

$$C_{\Sigma} \leq C$$

де C_{Σ} – сумарні витрати на реалізацію комплексу ЗЗ;

$C_{\text{доп}}$ – виділені на реалізацію комплексу грошові ресурси.

При цьому C_{Σ} обчислюється за допомогою наступного вираження:

$$C_{\Sigma} = \sum_{S} \left(\sum i_S + \sum C^{\text{с}}_{j_S} + \sum C^{\text{в}}_{k_S} + C_{\text{сегм}} \right) + C_{\text{пр}},$$

де S – число мережних сегментів;

$C^{\text{Б}}_{i_S}$ – вартість набору засобів захисту хоста, на якому обробляється інформація базового рівня критичності;

$C^{\text{С}}_{j_S}$ – вартість набору засобів захисту хоста, на якому обробляється інформація середнього рівня критичності;

$C^{\text{В}}_{k_S}$ – вартість набору засобів захисту хоста, на якому обробляється інформація високого рівня критичності;

$C_{\text{сегм}_S}$ – вартість набору засобів захисту на границі s-го мережного сегмента;

$C_{\text{пр}}$ – вартість наборів засобів захисту периметра.

Вибір комплексу засобів захисту для СЗІ досягається ітераційно шляхом наближення до раціонального состава, що задовольняє вимогам до припустимих витрат на його реалізацію.

У системі інтелектуальної підтримки раціональні рішення пропонується вибирати на основі використання експертних знань; у ній реалізується механізм придбання знань у процесі заповнення полів знань експертом при взаємодії його з

автоматизованою системою, виконується сукупність процедур над проблемною областю з використанням багатокритеріального порівняльного аналізу для виявлення в заданому експертом множини підмножини найкращих за критеріями переваги варіантів наборів, з яких формується раціональний комплекс засобів захисту.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Налаштування ПЗ.

– Контроль доступу до мережі на основі технології Cisco NAC.

– Повний перебір варіантів з сумісних засобів захисту.

– Завдання цільової функції.

– Розрахунок показників «захищеність» та «витрати».

– Обробка допоміжних матриць.

– Формування морфологічних матриць засобів захисту.

– Моніторинг мережі.

– БД засобів захисту.

– Формування критеріїв по показнику «захищеність».

– Формування критеріїв по показнику «витрати».

– БД функцій Cisco NAC.

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

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

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

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

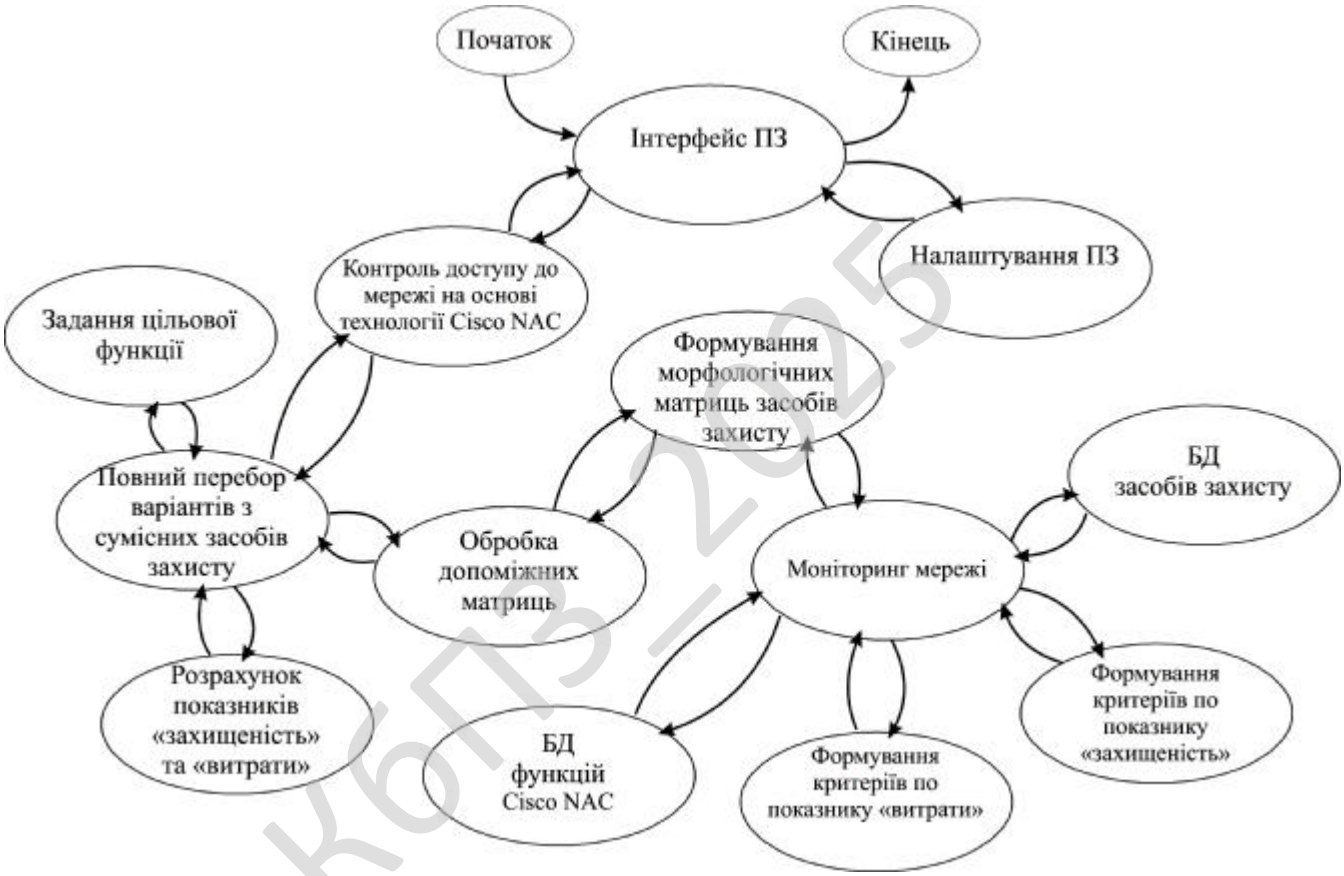

Рисунок 3.3 – Діаграма взаємодії процесів

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

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З якої 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.

Крім цього 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.

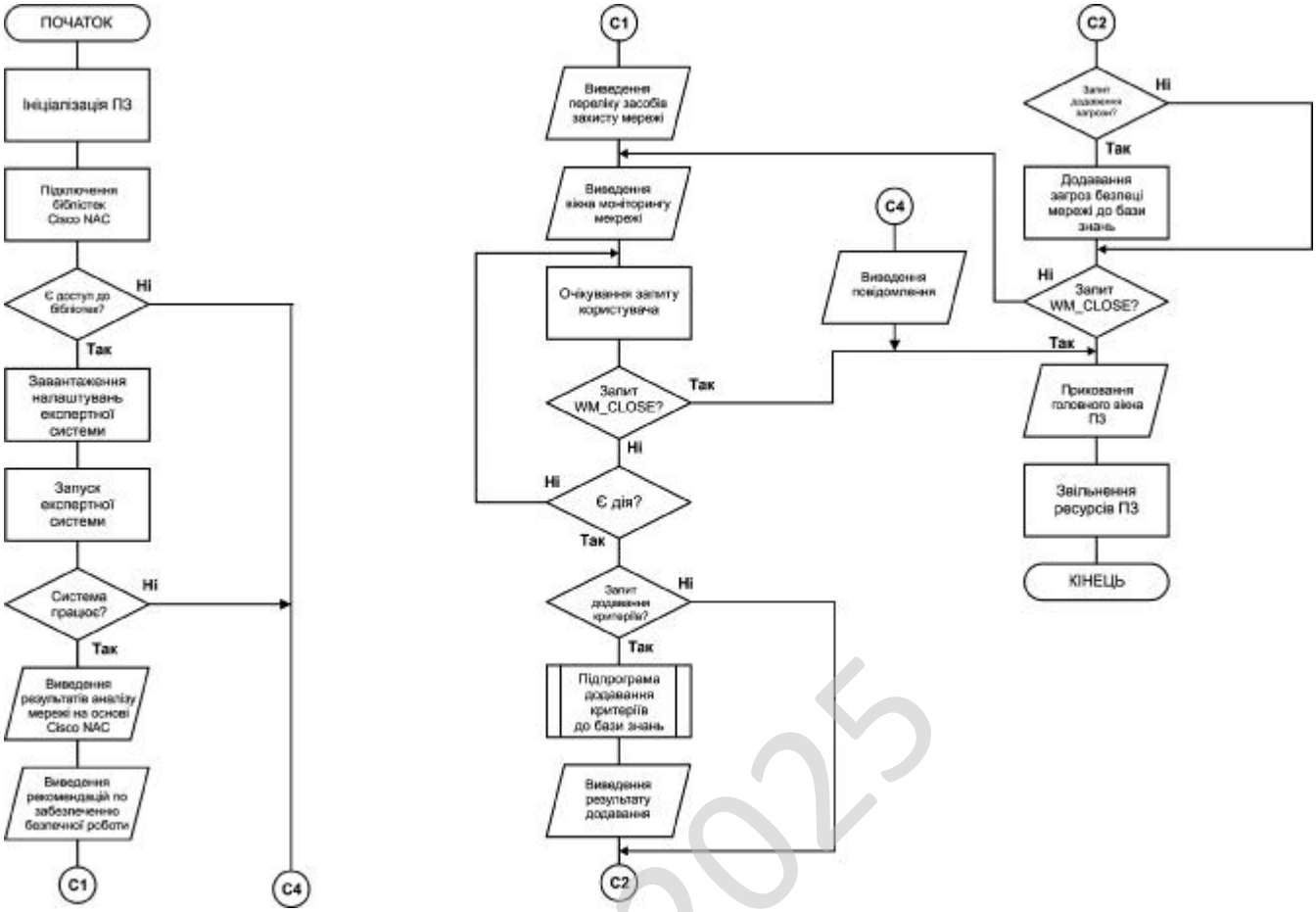

Рисунок 4.1 – Блок схема основної програми

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.

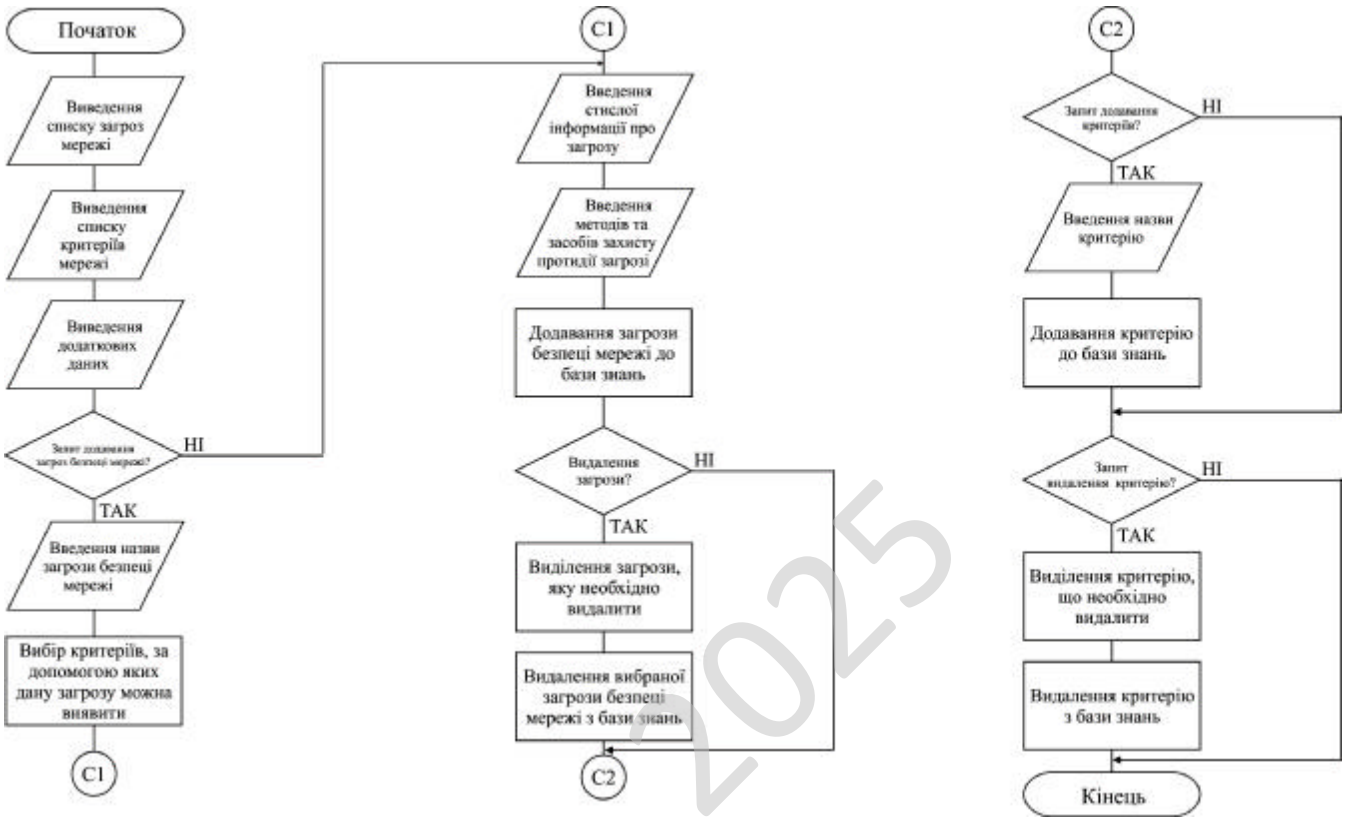

Рисунок 4.2 – Блок схема підпрограми

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.

Опис алгоритмів функціонування системи

Опишемо оцінювання рівня захищеності інформації який здійснюється на основі одного з основних положень уніфікованої концепції захисту – вимоги науково обґрунтованого підходу до оцінки (бажано в кількісному вираженні) необхідного рівня захищеності при проектуванні й у процесі експлуатації СЗІ. В процесі аналізу й оцінювання ризиків установлюється ступінь адекватності використовуваних або планованих наборів засобів захисту (ЗЗ) існуючим погрозам. Властивість «захищеність інформації» кожного ЗЗ, що входить у СЗІ, у сукупності визначає захищеність інформації в СЗІ в цілому.

Наявність уразливості ЗЗ може привести до порушення захищеності, тобто здійсненню погрози, тому при рішенні завдань захисту інформації першорядне значення має кількісна оцінка вразливостей засобів захисту. Оскільки вплив на інформацію різних деструктивних факторів значною мірою є випадковим, то як кількісну міру уразливості найбільше доцільно застосувати ймовірність порушення захищеності інформації. Неясність способу визначення значень імовірностей погроз і уразливістей є основною проблемою при одержанні кількісної оцінки ризику порушення інформаційної безпеки.

Відомо, що застосування методів класичної теорії ймовірностей припустимо при повторюваності дослідів і однаковості умов. Ця вимога в складних системах, якими є СЗІ, звичайно не виконується.

Відповідно до одному із принципів системного аналізу – принципу невизначеності, у процесі дослідження системи необхідний облік невизначеностей і випадків. Оскільки складні відкриті системи не підкоряються імовірнісним законам, у них варто оцінити найгірші ситуації відповідно до методу гарантованого результату, що пропонується використовувати при оцінці ймовірностей погроз.

Приймається, що значення показника *m*-го ЗЗ захищеність інформації $P_{бm}$ – це суб'єктивна ймовірність виявлення й блокування засобом захисту несанкціонованих дій, тобто теоретична очікувана ефективність бар'єра.

Очевидно, що ймовірність порушення захищеності $P^{н}_{бm}$ доповнює $P_{бm}$ до одиниці, тобто:

$$P^{н}_{бm} = 1 - P_{бm},$$

де $P^{н}_{бm}$ – імовірність порушення захищеності інформації, або ймовірність уразливості *m*-го ЗЗ (імовірність подолання бар'єра).

Пропонується ймовірносно-статичний підхід, при якому не враховується динаміка зміни значень імовірностей погроз і уразливістей у часі, оцінюються апріорні очікувані значення ймовірностей порушення захищеності інформації.

Особливістю пропонованого в магістерській роботі підходу є одержання чисельних значень суб'єктивних імовірностей на основі використання як приватні показники захищеності технічних характеристик і можливостей засобів захисту, декларуємих розроблювачами. Вирішується завдання одержання чисельної оцінки узагальненого показника якості засобу захисту.

Пропонується для одержання чисельної оцінки узагальненого показника якості засобу захисту захищеність інформації використовувати теорію нечітких множин. Для оцінки засобів захисту за кожним критерієм нижнього ієрархічного рівня формуються функції приналежності. При цьому використовуються методи побудови функцій приналежності, засновані на формалізації й інтеграції нечітких даних, сформованих експертом у процесі оцінювання параметрів реальних засобів захисту. Формулюються відповідні продукційні правила, що дозволяють

обробляти складні з'єднання. Достоїнство способу – відносно висока об'єктивність.

Метод оцінювання рівня захищеності інформації базується на трьохрубіжній моделі захисту, він розроблений для об'єкта захисту, архітектура якого відповідає основним принципам безпеки, що рекомендуються.

Відомо, що рівень захищеності й відносний ризик доповнюють один одного до одиниці. Пропонується розраховувати рівень захищеності η за формулою:

$$\eta = 1 - \overline{R} = 1 - \sum_s \frac{C_s}{C_\Sigma} \cdot P_s,$$

де $\overline{R}$ – відносний ризик;

C_s – частка вартості інформаційних ресурсів, що захищаються, у сегменті s;

s – номер сегмента;

S – число сегментів;

P_s – результуюча ймовірність погроз інформаційному середовищу сегмента;

C_Σ – сумарний неприйнятний збиток;

$\frac{C_s}{C_\Sigma}$ – коефіцієнт небезпеки сукупності погроз в s-ому сегменті, обумовлений як частка вартості інформації, що захищається, об'єкта захисту, оброблюваного в сегменті.

Таким чином, для оцінки рівня захищеності потрібна кількісна оцінка ймовірностей реалізації каналів несанкціонованого доступу.

Для оцінки ймовірності порушення захищеності підмножиною порушників $\{K'\}$ по підмножині можливих каналів несанкціонованого одержання інформації $\{J'\}$ для сегмента s використовується співвідношення:

$$P_{s\{J'\}\{K'\}} = 1 - \prod_{J'} \left(1 - P_{sjk}^{(б)}\right) \prod_{K'} \left(1 - P_{sjk}^{(б)}\right),$$

у якому приймається:

$$P_{sjk}^{(б)внш} \subset P_{sjk}^{(б)}, P_{sjk}^{(б)вн} \subset P_{sjk}^{(б)},$$

де $P_{sjk}^{(б)вн}, P_{sjk}^{(б)внш}$ – імовірність несанкціонованого одержання інформації, оброблюваної в *s*-му сегменті, відповідно, внутрішнім і зовнішнім порушником (зловмисником) для об'єкта захисту, що має точки виходу в глобальну мережу, зовнішні виділені канали зв'язку, для якого можливі віддалені атаки через периметр.

З обліком прийнятої трьохрубіжної моделі захисту $P_{sjk}^{(б)внш}$ обчислюється за формулою:

$$P_{sjk}^{(б)внш} = 1 - \prod_{l=1}^{3} (1 - P_{sjk\ l}^{внш}),$$

де $P_{sjk\ l}^{внш}$ – імовірність несанкціонованого одержання інформації, оброблюваної в *s*-му сегменті, зловмисника, зовнішнього порушника у випадку подолання відповідного рубежу захисту *l*.

Імовірність $P_{sjk\ l}^{внш}$ залежить від чотирьох факторів і визначається залежністю:

$$P_{sjkl}^{внш} = P_{skl}^{д} \cdot P_{sjkl}^{н} \cdot P_{sjl}^{к} \cdot P_{sjl}^{и},$$

де $P_{sk\ l}^{д}$ – імовірність спроби доступу зловмисника або зовнішнього порушника-користувача до *l*-му рубежу захисту;

$P_{sjk\ l}^{н}$ – імовірність подолання зловмисником або зовнішнім порушником *l*-го рубежу захисту;

$P_{sj\ l}^{к}$ – імовірність наявності трафіка із сегмента s через l-*й* рубіж захисту, залежить від технології обробки інформації на об'єкті захисту, імовірність можна прийняти рівній частоті роботи каналу;

$P_{sj\ l}^{и}$ – імовірність наявності інформації, що захищається, *s*-го сегмента в трафіку в момент подолання зовнішнім порушником *l*-го рубежу захисту, залежить від технології обробки інформації на об'єкті захисту.

Внутрішній порушник у процесі реалізації каналів несанкціонованого доступу повинен перебороти два рубежі захисту.

Тоді ймовірність несанкціонованого одержання інформації, оброблюваної в сегменті s, внутрішнім порушником обчислюється за формулою:

$$P_{sj}^{(б)вн} = 1 - \prod_{l=1}^{2} (1 - P_{sjl}^{вн}),$$

де $P_{sl}^{вн}$ – імовірність несанкціонованого доступу до інформації, оброблюваної в *s*-му сегменті, внутрішнього порушника у випадку подолання відповідного рубежу захисту *l*.

З перерахованих імовірностей, що входять у формули для розрахунку $P_{sjl}^{вн}$ й $P_{sjkl}^{внш}$, одна з імовірностей, а саме $P_{sjkl}^{н}$, залежить від якості використовуваних у системі засобів захисту й кількості бар'єрів на рубежі захисту. Якщо порушникові необхідно перебороти *M* бар'єрів на рубежі захисту, то ймовірність його вдалої атаки визначається як добуток:

$$P_{sjkl}^{н} = \prod_{m=1}^{M} P_{бm}^{н} = \prod_{m=1}^{M} (1 - P_{бm}).$$

На основі пропонованого методу оцінки ризику порушення інформаційної безпеки розробляються алгоритм і функціональна модель прогнозування рівня захищеності інформації за допомогою IDEF 0-технології.

Автоматизація прийняття рішень по управлінню ЗІ відповідає високому рівню моделі зрілості процесів управління, забезпечує обґрунтованість і раціональність рішень на основі використання математичного апарата, знижує працєвитрати на виконання обчислень.

Розроблено інструментальні програмні комплекси для автоматизованої системи інтелектуальної підтримки організаційно – технічного й оперативного управління ЗІ.

На основі програмного засобу «Прийняття рішень в умовах ризику», що реалізує метод вибору раціонального варіанта реагування на події безпеки,

розроблений модуль «Оперативний вирішувач» системи інтелектуальної підтримки оперативного управління. Результати моделювання роботи модуля наведені в таблиці 4.1.

Таблиця 4.1 – Результати моделювання роботи модуля «Оперативний вирішувач»

Види погроз	Параметри інформаційного середовища / Варіанти реагування		
Локальне мережне вторгнення порушника	$A = 2, B = 3, P_a = 0{,}57$	$A = 1, B = 1, P_a = 0{,}238$	
	Завершення сесії з атакуючим вузлом	Відсилання попередження користувачеві	
Зовнішнє вторгнення по радіоканалу (Wi Fi, Wi Max)	$A = 3, C = 3,$ $P_a = 0{,}678$	$A = 1, З = 2, P_a = 0{,}4$	$A = 1, З = 1, P_a = 0{,}3$
	Блокування ТД	DOS-атака на атакуючу станцію	Відсутність реагування
Віддалена атака через периметр по лінії зв'язку	$A = 3, B = 4, C = 2,$ $P_a = 0{,}742$	$A = 1, B = 1, C = 1,$ $P_a = 0{,}238$	$A = 1, P_a = 0{,}08$
	Блокування доступу до сервісу в мережі	Переконфігурування сервісів безпеки з метою блокування IP	Відсилання попередження на IP-адресу

Досвід розробки, впровадження й супроводи інформаційної системи показує, що якщо вона забезпечує бізнес-процеси досить великої організації, те практично ніколи не буває статичної. Як наслідок, рішення по безпеці, прийняті при проектуванні СЗІ, дуже швидко втрачають відповідність тій системі, ресурси якого вони покликані захищати. Тому в такій системі важливим є підтримка заданого рівня захищеності в процесі експлуатації постійно, що змінюється системи. Інструментальний програмний комплекс «Система підтримки прийняття

рішень по управлінню захистом інформації на об'єкті інформатизації», призначений для обґрунтованого вибору раціонального комплексу засобів захисту при проектуванні СЗІ й у ході планування в процесі експлуатації, розроблений для системи інтелектуальної підтримки організаційно – технічного управління ЗІ.

Для реалізації механізму придбання знань у рамках розробленого програмного комплексу організується взаємодію експерта з автоматизованою системою, у процесі якого експерт заповнює запропоновані йому розроблені поля знань. У плановому вирішувачі реалізований алгоритм пошуку шляхів від вхідних даних до вихідних – раціональним набором засобів захисту для заданих рубежів. Створений програмний продукт дозволяє не тільки одержати результат на основі уведених даних і знань, але й увести базу знань (створення, динамічне розширення, повторне використання).

Програмний комплекс «Система підтримки прийняття рішень по управлінню захистом інформації на об'єкті інформатизації» був використаний для рішення завдання забезпечення інформаційної безпеки в мережі транспортного підприємства – сегменті корпоративної інформаційної системи.

Необхідність модернізації СЗІ виникла у зв'язку з недостатньою пропускною здатністю віртуальних каналів VPN, побудованих на базі програмного комплексу VIPNet Custrom 3. Для збільшення швидкості обміну інформацією до 40 Мбіт/с у зв'язку зі зрослим обсягом трафіка, переданого по корпоративних каналах, розглядається можливість і економічна доцільність переходу із програмного рішення VPN на програмно – апаратний модуль NME – RVPN VIPNet (S – Terra CSP і Інфотекс) для маршрутизаторів Cisco 2811, що реалізує апаратне шифрування трафіка зі швидкістю 40 Мбіт/с.

Дане рішення, у зв'язку з необхідністю заміни маршрутизаторів, з одного боку, зажадає значних витрат, а з іншого боку – не приведе до комплексного рішення ЗІ. За допомогою програмного комплексу «Система підтримки прийняття рішень по управлінню захистом інформації на об'єкті інформатизації» вирішене

завдання вибору раціонального модульного состава СЗІ для використання наявних у мережі маршрутизаторів Cisco (їхня заміна не буде потрібна).

Комплекс засобів захисту містить у собі сумісні з наявною апаратурою сертифіковані ДСТЗІ СБУ програмно – апаратні засоби вітчизняних і закордонних виробників.

Застосування запропонованого підходу до побудови комплексної СЗІ дозволяє зменшити витрати на захист на 44% у порівнянні з альтернативним комплексом, що забезпечує той же клас захищеності інформаційної системи 1Г.

Розроблений програмний комплекс для одержання чисельної оцінки рівня захищеності (ризику порушення інформаційної безпеки) «Розрахунок чисельного значення ризику порушення інформаційної безпеки (InfoRisk)» який дозволяє оцінити рівень захищеності на технічному рівні забезпечення ІЗ із використанням моделі погроз.

Програмний засіб має наступні можливості: дозволяє оцінювати рівень захищеності інформації на об'єкті захисту, що складається з множини сегментів, у яких обробляється інформація різного рівня критичності; дозволяє задавати вихідні дані по кількості сегментів, застосовуваних або планованих засобів захисту, по рівнях критичності інформаційних ресурсів; застосуємих на стадії розробки СЗІ й на стадії експлуатації системи; забезпечує оперативність оцінювання; дозволяє проводити порівняльний аналіз різних комплексів засобів захисту в ході управління ризиками; дозволяє однозначно враховувати специфіку функціонування конкретного об'єкта захисту, реальні погрози для конкретних ключових ресурсів; розрахунок ризику проводиться з мінімальним залученням експертів.

Знання експерта використовуються лише для побудови функцій приналежності й складаня продукційних правил при використанні інструментарію нечіткої логіки Fuzzy Toolbox програмного продукту Matlab, у якому обчислюються показники ЗЗ «захищеність інформації». Оцінювання даного показника здійснюється на основі використання відомостей про технічні

характеристики засобів захисту.

Показник ризику формується на основі заданих значень цінностей потребуючі захисти ресурсів. Адекватність методу оцінювання ризику порушення ІБ, реалізованого в програмному комплексі InfoRisk, не залежить від наявності або відсутності достовірних статистичних даних по інцидентах ІБ, тим більше, що на етапі проектування СЗІ такі дані відсутні.

Наведено результати розрахунку прогнозованого ризику в СЗІ з раціональним модульним составом і до модернізації системи захисту. При проведенні розрахунків ураховувалася можлива наявність зловмисника, що реалізує віддалене вторгнення через периметр, наявність зовнішніх і внутрішніх користувачів – порушників і інсайдера, що має високі привілеї й порушує політику безпеки. Розрахункове прогнозоване значення ризику склало 1,84%, що в 6 разів менше значення ризику в існуючої СЗІ. Розрахунок проводився на основі методу гарантованого результату, для найгіршої ситуації.

У цілому, на основі проведених досліджень можна констатувати ефективність використаного системного підходу й запропонованих моделей, методів і алгоритмів для управління ЗІ у сегменті корпоративної інформаційної системи.

Система контролю доступу у мережу на основі технології Cisco NAC є комплексним рішенням, що забезпечує ідентифікацію, автентифікацію та авторизацію користувачів, які підключаються до мережі. Вона базується на принципах Network Access Control (NAC) та включає кілька основних компонентів.

Основні компоненти системи

1. Сервер автентифікації та контролю доступу. Використовується для перевірки користувачів перед наданням доступу до ресурсів мережі.

2. Клієнтський агент NAC. Відповідає за відправку облікових даних та інформації про стан пристрою на сервер контролю доступу.

3. Система моніторингу та логування. Записує всі спроби підключення, успішні та невдалі спроби автентифікації.

4. Механізм динамічного керування доступом. Застосовує політики безпеки на основі отриманих даних про користувача та його пристрій.

Архітектура системи складається з кількох рівнів.

Основний рівень - сервер NAC, який приймає запити на підключення та обробляє їх. Наступний рівень - клієнтські пристрої, що взаємодіють із сервером для автентифікації. Третій рівень - адміністраторський інтерфейс, який дозволяє переглядати логі, налаштовувати правила доступу та контролювати статус підключених пристроїв

Розрахунки продуктивності

Для оцінки продуктивності використовується модель навантаження, де враховується кількість користувачів, що підключаються одночасно.

Припустимо, що система обслуговує 1000 користувачів, середній час автентифікації складає 0.5 секунди. Отже, загальне навантаження на сервер у пікові періоди можна розрахувати як

1000 * 0.5 = 500 секунд (загальний час перевірки).

Якщо сервер здатний обробляти 50 запитів на секунду, то необхідний серверний ресурс для пікових навантажень розраховується як

1000 / 50 = 20 секунд (загальний час обробки запитів у піковому режимі).

Таким чином на основі цих даних вибирається апаратна конфігурація сервера.

Частина коду реалізації системи контролю доступу на Python

```
import time
from typing import Dict

class NACServer:
    def __init__(self):
        self.users_db = {
            "user1": "password1",
            "user2": "password2"
        }
```

```
            self.access_logs = []

        def authenticate(self, username: str, password: str) -> bool:
            if username in self.users_db and self.users_db[username] ==
password:
                self.access_logs.append(f"{username} - Доступ дозволено")
                return True
            else:
                self.access_logs.append(f"{username} - Доступ заборонено")
                return False

        def log_access_attempts(self):
            for log in self.access_logs:
                print(log)

    class NACClient:
        def __init__(self, server: NACServer, username: str, password: str):
            self.server = server
            self.username = username
            self.password = password

        def request_access(self):
            if self.server.authenticate(self.username, self.password):
                print("Доступ дозволено")
            else:
                print("Доступ заборонено")

    # Імітація підключення клієнтів
    server = NACServer()
    clients = [
        NACClient(server, "user1", "password1"),
        NACClient(server, "user2", "wrongpassword"),
        NACClient(server, "user3", "password3")
    ]

    for client in clients:
        client.request_access()
        time.sleep(1)

    server.log_access_attempts()
```

Таким чином система контролю доступу на основі Cisco NAC дозволяє ефективно управляти підключеннями до мережі. Вона забезпечує ідентифікацію користувачів, облік підключень та застосування політик безпеки. Реалізація на Python демонструє основні принципи роботи такої системи та може бути розширена для інтеграції з реальними мережевими пристроями.

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом.

Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму DES.

DES є класичною мережею Фейштеля із двома гілками. Дані шифруються 64-бітними блоками, використовуючи 56-бітний ключ. Алгоритм перетворить за кілька раундів 64-бітний вхід в 64-бітний вихід. Довжина ключа дорівнює 56 бітам. Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти переставляються у відповідності зі стандартною

таблицею. Наступний етап складається з 16 раундів однієї й тої ж функції, що використовує операції зрушення й підстановки. На третьому етапі ліва й права половини виходу останньої (16-й) ітерації міняються місцями. Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

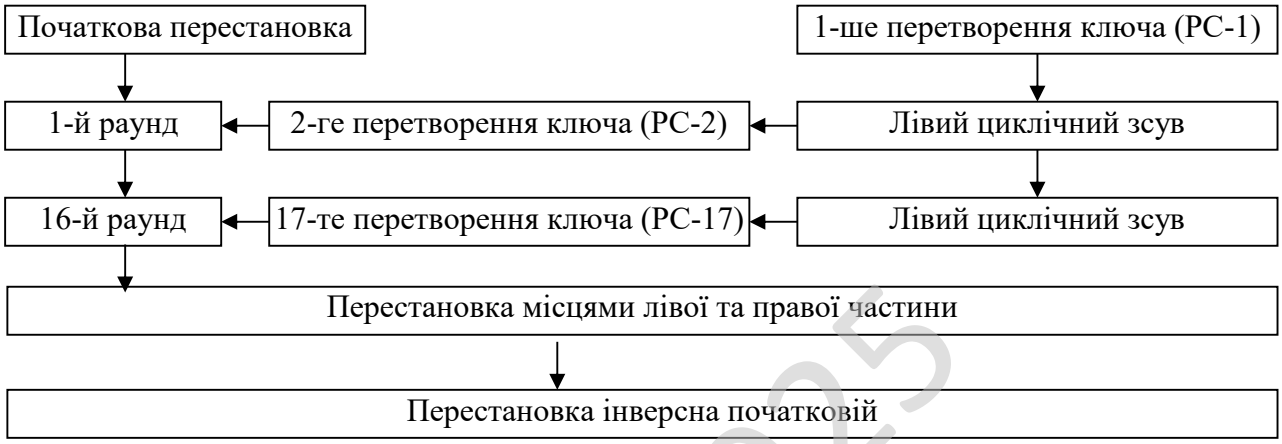

Рисунок 4.3 – Загальна схема DES

Праворуч на рисунку показаний спосіб, яким використовується 56-бітний ключ. Спочатку ключ подається на вхід функції перестановки. Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зрушення й перестановки. Функція перестановки та сама для кожного раунду, але підключи K_i для кожного раунду виходять різні внаслідок повторюваного зрушення біт ключа.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Розділ критеріїв до бази знань.

– Розділ результатів.

– Розділ додаткова інформація.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

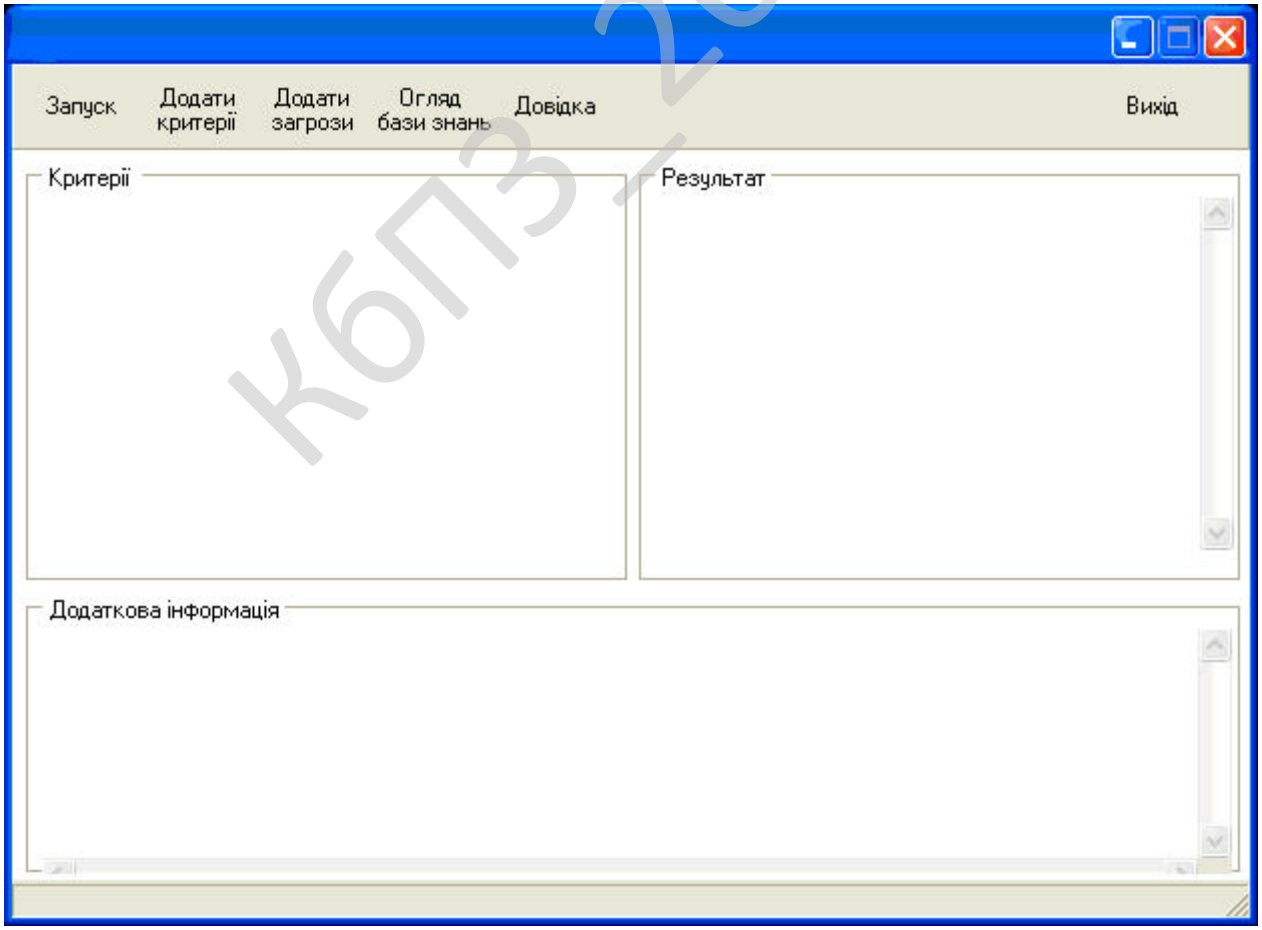

Рисунок 5.1 – Основне вікно програми

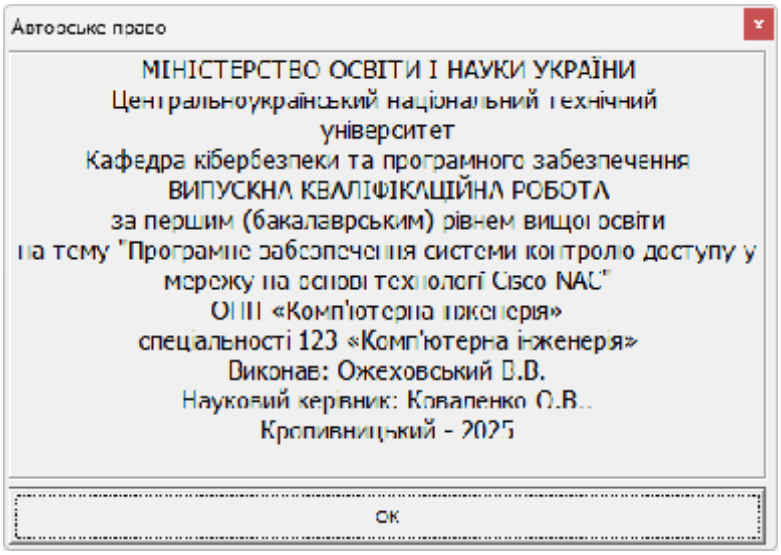

Рисунок 5.2 – Авторське право

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

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

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи контролю доступу у мережу на основі технології Cisco NAC.

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 доступу у мережу на основі технології Cisco NAC.

– Досліджена система контролю доступу у мережу на основі технології Cisco NAC.

– На основі отриманих результатів досліджень створена програмна реалізація системи контролю доступу у мережу на основі технології Cisco NAC.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання контролю доступу у мережу на основі технології Cisco NAC.

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 доступу у мережу на основі технології Cisco NAC. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід,

зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм DES.

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. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

2. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

3. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

4. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

5. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

6. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

7. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

8. 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.

9. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

10. 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

11. 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.

12. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

13. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

14. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

15. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

16. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing*

Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418

17. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

18. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

19. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

20. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

22. 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.

23. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

24. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

25. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

26. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

27. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

28. 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.

29. 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.

30. 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.

31. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

32. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

33. 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.

34. 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.

35. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

36. 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.

37. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

38. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced*

Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18 - 21 September 2019. P.713-718.

39. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

40. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

41. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

42. 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.

43. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

44. 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.

45. 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.

46. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

47. 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.

48. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

49. 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.

50. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об’єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №*3*(19), 2023, С. 176-196.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи контролю доступу у мережу на основі технології Cisco NAC.

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. № 46-02 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 доступу у мережу на основі технології Cisco NAC.

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 доступу у мережу на основі технології Cisco NAC;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

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 – 78 аркушів.

8 Етапи розробки

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

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

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

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 2.06.2025 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

_____ Коваленко О.В.

Програмне забезпечення системи контролю доступу у мережу на основі технології Cisco NAC

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
#Ініціалізація модулів, необхідних для роботи системи
import time
import random
import hashlib
import sys

#Оголошення класу Device для представлення пристроїв
class Device:
#Ініціалізація пристрою з унікальним ID, MAC-адресою, IP-адресою та власником
    def __init__(self, device_id, mac_address, ip_address, owner_username):
#Присвоєння унікального ідентифікатора пристрою
        self.device_id = device_id
#Присвоєння MAC-адреси пристрою
        self.mac_address = mac_address
#Присвоєння IP-адреси пристрою
        self.ip_address = ip_address
#Присвоєння власника пристрою
        self.owner_username = owner_username
#Встановлення статусу автентифікації пристрою на False
        self.authenticated = False
#Встановлення статусу карантину на False
        self.quarantine = False
#Ініціалізація часу останнього сканування пристрою
        self.last_scanned = None
#Метод для оновлення IP-адреси пристрою
    def update_ip(self, new_ip):
#Присвоєння нового значення IP-адреси
        self.ip_address = new_ip
#Метод для відмітки пристрою як автентифікованого
    def mark_authenticated(self):
#Встановлення прапорця автентифікації в True
        self.authenticated = True
#Метод для відмітки пристрою як знаходиться у карантині
    def mark_quarantine(self):
#Встановлення прапорця карантину в True
        self.quarantine = True
#Метод для зняття пристрою з карантину
    def clear_quarantine(self):
#Встановлення прапорця карантину в False
        self.quarantine = False
#Метод повернення рядкового представлення об’єкта Device
    def __str__(self):
#Повернення рядкового опису пристрою
        return ("Device ID: " + self.device_id + ", MAC: " + self.mac_address +
", IP: " + self.ip_address +
                ", Owner: " + self.owner_username + ", Authenticated: " +
str(self.authenticated) +
                ", Quarantine: " + str(self.quarantine))

#Оголошення класу User для представлення користувачів системи
class User:
#Ініціалізація користувача з ім'ям користувача та паролем
    def __init__(self, username, password):
#Присвоєння імені користувача
        self.username = username
#Хешування пароля користувача для безпечного зберігання
        self.password_hash = generate_hash(password)
#Метод для перевірки правильності пароля
    def verify_password(self, password):
#Порівняння згенерованого хеша вхідного пароля з збереженим хешем
        return generate_hash(password) == self.password_hash
#Метод повернення рядкового представлення об’єкта User
    def __str__(self):
#Повернення рядкового опису користувача
        return "User: " + self.username
```

```
#Оголошення класу NetworkPolicy для визначення мережевих політик
class NetworkPolicy:
#Ініціалізація політики з унікальним ID, описом, дозволеними користувачами та
пристроями
    def __init__(self, policy_id, description, allowed_users, allowed_devices):
#Присвоєння унікального ідентифікатора політики
        self.policy_id = policy_id
#Присвоєння опису політики
        self.description = description
#Збереження списку дозволених користувачів
        self.allowed_users = allowed_users
#Збереження списку дозволених пристроїв
        self.allowed_devices = allowed_devices
#Метод для додавання користувача до списку дозволених
    def add_allowed_user(self, username):
#Перевірка, чи користувач відсутній у списку, і додавання у разі потреби
        if username not in self.allowed_users:
            self.allowed_users.append(username)
#Метод для додавання пристрою до списку дозволених
    def add_allowed_device(self, device_id):
#Перевірка, чи пристрій відсутній у списку, і додавання у разі потреби
        if device_id not in self.allowed_devices:
            self.allowed_devices.append(device_id)
#Метод для видалення користувача зі списку дозволених
    def remove_allowed_user(self, username):
#Перевірка наявності користувача у списку та видалення, якщо він існує
        if username in self.allowed_users:
            self.allowed_users.remove(username)
#Метод для видалення пристрою зі списку дозволених
    def remove_allowed_device(self, device_id):
#Перевірка наявності пристрою у списку та видалення, якщо він існує
        if device_id in self.allowed_devices:
            self.allowed_devices.remove(device_id)
#Метод повернення рядкового представлення об’єкта NetworkPolicy
    def __str__(self):
#Повернення рядкового опису політики з усіма деталями
        return ("Policy ID: " + self.policy_id + ", Description: " +
self.description +
                ", Allowed Users: " + str(self.allowed_users) + ", Allowed
Devices: " + str(self.allowed_devices))

#Оголошення головного класу CiscoNACController, який управляє всіма компонентами
системи
class CiscoNACController:
#Ініціалізація контролера з порожніми словниками для користувачів, пристроїв і
політик
    def __init__(self):
#Ініціалізація словника пристроїв
        self.devices = {}
#Ініціалізація словника користувачів
        self.users = {}
#Ініціалізація словника політик
        self.policies = {}
#Ініціалізація списку для зберігання логів подій
        self.events_log = []
#Метод для логування подій з міткою часу
    def log_event(self, event):
#Отримання поточної дати і часу
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
#Формування рядка логування події
        log_entry = timestamp + " - " + event
#Додавання події до списку логів
        self.events_log.append(log_entry)
#Виведення повідомлення про подію на екран
        print(log_entry)
#Метод для реєстрації нового користувача
    def register_user(self, username, password):
#Перевірка наявності користувача в системі
        if username in self.users:
```

```
            self.log_event("Спроба зареєструвати існуючого користувача " +
username)
            return False
#Створення нового об’єкта User
        new_user = User(username, password)
#Додавання користувача до словника користувачів
        self.users[username] = new_user
#Логування успішної реєстрації користувача
        self.log_event("Користувача " + username + " зареєстровано успішно")
        return True
#Метод для реєстрації нового пристрою
    def register_device(self, device_id, mac_address, ip_address,
owner_username):
#Перевірка наявності пристрою в системі
        if device_id in self.devices:
            self.log_event("Спроба зареєструвати існуючий пристрій " +
device_id)
            return False
#Перевірка існування власника пристрою
        if owner_username not in self.users:
            self.log_event("Спроба зареєструвати пристрій " + device_id + " для
неіснуючого користувача " + owner_username)
            return False
#Створення нового об’єкта Device
        new_device = Device(device_id, mac_address, ip_address, owner_username)
#Додавання пристрою до словника пристроїв
        self.devices[device_id] = new_device
#Логування успішної реєстрації пристрою
        self.log_event("Пристрій " + device_id + " зареєстровано успішно для
користувача " + owner_username)
        return True
#Метод для автентифікації пристрою на основі пароля власника
    def authenticate_device(self, device_id, password):
#Перевірка наявності пристрою в системі
        if device_id not in self.devices:
            self.log_event("Аутентифікація не вдалася: пристрій " + device_id +
" не знайдено")
            return False
#Отримання об’єкта пристрою
        device = self.devices[device_id]
#Отримання імені власника пристрою
        owner_username = device.owner_username
#Перевірка наявності власника пристрою в системі
        if owner_username not in self.users:
            self.log_event("Аутентифікація не вдалася: власник " +
owner_username + " не знайдений для пристрою " + device_id)
            return False
#Отримання об’єкта користувача
        user = self.users[owner_username]
#Перевірка правильності введеного пароля
        if not user.verify_password(password):
            self.log_event("Аутентифікація не вдалася: неправильний пароль для
користувача " + owner_username)
            return False
#Встановлення статусу автентифікації пристрою
        device.mark_authenticated()
#Логування успішної автентифікації
        self.log_event("Пристрій " + device_id + " успішно аутентифіковано")
        return True
#Метод для застосування політики мережевого доступу до пристрою
    def enforce_policy(self, device_id, policy_id):
#Перевірка наявності політики в системі
        if policy_id not in self.policies:
            self.log_event("Застосування політики не вдалося: політику " +
policy_id + " не знайдено")
            return False
#Перевірка наявності пристрою в системі
        if device_id not in self.devices:
```

```
            self.log_event("Застосування політики не вдалося: пристрій " +
device_id + " не знайдено")
            return False
#Отримання об’єкта політики
        policy = self.policies[policy_id]
#Отримання об’єкта пристрою
        device = self.devices[device_id]
#Перевірка відповідності власника пристрою або самого пристрою політиці
        if device.owner_username in policy.allowed_users or device.device_id in
policy.allowed_devices:
            self.log_event("Пристрій " + device_id + " відповідає політиці " +
policy_id)
            return True
        else:
            self.log_event("Пристрій " + device_id + " не відповідає політиці "
+ policy_id + ", застосовано карантин")
            device.mark_quarantine()
            return False
#Метод для додавання нової політики в систему
    def add_policy(self, policy_id, description, allowed_users,
allowed_devices):
#Перевірка наявності політики з таким ID
        if policy_id in self.policies:
            self.log_event("Спроба додати існуючу політику " + policy_id)
            return False
#Створення нового об’єкта політики
        new_policy = NetworkPolicy(policy_id, description, allowed_users,
allowed_devices)
#Додавання політики до словника політик
        self.policies[policy_id] = new_policy
#Логування успішного додавання політики
        self.log_event("Політику " + policy_id + " додано успішно")
        return True
#Метод для проведення безпекового сканування пристрою
    def scan_device_security(self, device_id):
#Перевірка наявності пристрою для сканування
        if device_id not in self.devices:
            self.log_event("Сканування не вдалося: пристрій " + device_id + " не
знайдено")
            return False
#Отримання об’єкта пристрою
        device = self.devices[device_id]
#Симуляція результату сканування шляхом випадкового вибору статусу
        scan_result = random.choice(["secure", "vulnerable"])
#Запис часу останнього сканування пристрою
        device.last_scanned = time.strftime("%Y-%m-%d %H:%M:%S",
time.localtime())
#Перевірка результату сканування
        if scan_result == "vulnerable":
            self.log_event("Сканування безпеки: пристрій " + device_id + " має
вразливості")
            device.mark_quarantine()
            return False
        else:
            self.log_event("Сканування безпеки: пристрій " + device_id + " є
безпечним")
            device.clear_quarantine()
            return True
#Метод для оновлення IP-адреси пристрою
    def update_device_ip(self, device_id, new_ip):
#Перевірка наявності пристрою для оновлення
        if device_id not in self.devices:
            self.log_event("Оновлення IP не вдалося: пристрій " + device_id + "
не знайдено")
            return False
#Отримання об’єкта пристрою
        device = self.devices[device_id]
#Виклик методу оновлення IP-адреси пристрою
        device.update_ip(new_ip)
```

```
#Логування успішного оновлення IP-адреси
        self.log_event("Пристрій " + device_id + " оновив IP на " + new_ip)
        return True
#Метод для блокування пристрою в системі
    def block_device(self, device_id):
#Перевірка наявності пристрою для блокування
        if device_id not in self.devices:
            self.log_event("Блокування не вдалося: пристрій " + device_id + " не
знайдено")
            return False
#Отримання об’єкта пристрою
        device = self.devices[device_id]
#Виклик методу відмітки пристрою як заблокованого (у карантині)
        device.mark_quarantine()
#Логування події блокування пристрою
        self.log_event("Пристрій " + device_id + " заблоковано від доступу до
мережі")
        return True
#Метод для розблокування пристрою
    def unblock_device(self, device_id):
#Перевірка наявності пристрою для розблокування
        if device_id not in self.devices:
            self.log_event("Розблокування не вдалося: пристрій " + device_id + "
не знайдено")
            return False
#Отримання об’єкта пристрою
        device = self.devices[device_id]
#Виклик методу зняття карантину з пристрою
        device.clear_quarantine()
#Логування події розблокування пристрою
        self.log_event("Пристрій " + device_id + " розблоковано")
        return True
#Метод для отримання детального статусу пристрою
    def get_device_status(self, device_id):
#Перевірка наявності пристрою для отримання статусу
        if device_id not in self.devices:
            self.log_event("Отримання статусу не вдалося: пристрій " + device_id
+ " не знайдено")
            return "Пристрій не знайдено"
#Отримання об’єкта пристрою
        device = self.devices[device_id]
#Формування рядкового статусу пристрою
        status = str(device)
#Логування отримання статусу пристрою
        self.log_event("Статус пристрою " + device_id + ": " + status)
        return status
#Метод для запуску діагностики всієї мережі
    def run_diagnostic(self):
#Логування початку діагностики мережі
        self.log_event("Запуск діагностики мережі для всіх пристроїв")
#Проходження по кожному зареєстрованому пристрою для проведення сканування
        for device_id in self.devices:
            self.scan_device_security(device_id)
#Логування завершення діагностики мережі
        self.log_event("Діагностика мережі завершена")
#Метод для запиту детальної інформації про пристрій
    def query_device(self, device_id):
#Перевірка наявності пристрою для запиту
        if device_id not in self.devices:
            self.log_event("Запит не вдалося: пристрій " + device_id + " не
знайдено")
            return None
#Отримання об’єкта пристрою
        device = self.devices[device_id]
#Логування запиту інформації про пристрій
        self.log_event("Запит деталей пристрою " + device_id)
#Повернення словника з інформацією про пристрій
        return {
            "device_id": device.device_id,
```

```
            "mac_address": device.mac_address,
            "ip_address": device.ip_address,
            "owner": device.owner_username,
            "authenticated": device.authenticated,
            "quarantine": device.quarantine,
            "last_scanned": device.last_scanned
        }
#Метод для симуляції моніторингу мережевого трафіку
    def simulate_network_traffic(self):
#Логування початку симуляції мережевого трафіку
        self.log_event("Запуск симуляції мережевого трафіку")
#Циклічний перебір для симуляції передачі пакетів даних
        for i in range(1, 6):
            packet_info = "Пакет " + str(i) + ": " + str(random.randint(1000,
9999)) + " байт передано"
            self.log_event(packet_info)
            time.sleep(0.3)
#Логування завершення симуляції мережевого трафіку
        self.log_event("Симуляція мережевого трафіку завершена")
#Метод для оновлення існуючої політики з новими дозволеними користувачами або
пристроями
    def update_policy(self, policy_id, allowed_users=None,
allowed_devices=None):
#Перевірка наявності політики для оновлення
        if policy_id not in self.policies:
            self.log_event("Оновлення політики не вдалося: політику " +
policy_id + " не знайдено")
            return False
#Отримання об’єкта політики
        policy = self.policies[policy_id]
#Оновлення списку дозволених користувачів, якщо передано нові дані
        if allowed_users is not None:
            for user in allowed_users:
                policy.add_allowed_user(user)
            self.log_event("Політика " + policy_id + " оновлена новими
дозволеними користувачами")
#Оновлення списку дозволених пристроїв, якщо передано нові дані
        if allowed_devices is not None:
            for device in allowed_devices:
                policy.add_allowed_device(device)
            self.log_event("Політика " + policy_id + " оновлена новими
дозволеними пристроями")
        return True
#Метод для видалення існуючої політики з системи
    def remove_policy(self, policy_id):
#Перевірка наявності політики для видалення
        if policy_id not in self.policies:
            self.log_event("Видалення політики не вдалося: політику " +
policy_id + " не знайдено")
            return False
#Видалення політики зі словника політик
        del self.policies[policy_id]
#Логування успішного видалення політики
        self.log_event("Політику " + policy_id + " видалено успішно")
        return True
#Метод для відображення всіх збережених логів подій
    def display_all_logs(self):
#Логування запиту на відображення логів
        self.log_event("Відображення всіх системних логів")
#Виведення кожного лог-запису на екран
        for log_entry in self.events_log:
            print(log_entry)
#Метод для відображення всіх зареєстрованих пристроїв
    def display_all_devices(self):
#Логування запиту на відображення пристроїв
        self.log_event("Відображення всіх зареєстрованих пристроїв")
#Виведення інформації про кожен пристрій
        for device in self.devices.values():
            print(device)
```

```
#Метод для відображення всіх зареєстрованих користувачів
    def display_all_users(self):
#Логування запиту на відображення користувачів
        self.log_event("Відображення всіх зареєстрованих користувачів")
#Виведення інформації про кожного користувача
        for user in self.users.values():
            print(user)
#Метод для відображення всіх мережевих політик
    def display_all_policies(self):
#Логування запиту на відображення політик
        self.log_event("Відображення всіх мережевих політик")
#Виведення інформації про кожну політику
        for policy in self.policies.values():
            print(policy)
#Метод для симуляції аналізу мережевого трафіку з детальним виводом даних
    def simulate_traffic_analysis(self):
#Логування початку аналізу мережевого трафіку
        self.log_event("Початок детального аналізу мережевого трафіку")
#Симуляція аналізу шляхом генерації випадкових показників
        for j in range(3):
            analysis_detail = "Аналіз " + str(j+1) + ": затримка " +
str(random.uniform(0.1, 1.0)) + " сек, втрати " + str(random.randint(0, 5)) + "
пакетів"
            self.log_event(analysis_detail)
            time.sleep(0.4)
#Логування завершення аналізу мережевого трафіку
        self.log_event("Детальний аналіз мережевого трафіку завершено")
#Метод для симуляції виявлення вторгнень
    def simulate_intrusion_detection(self):
#Логування запуску системи виявлення вторгнень
        self.log_event("Запуск системи виявлення вторгнень")
#Симуляція декількох спроб вторгнення з випадковими результатами
        for k in range(4):
            intrusion_status = random.choice(["Нормальний трафік", "Підозріле
з'єднання", "Атака типу DoS", "Аномалія в мережі"])
            self.log_event("Подія " + str(k+1) + ": " + intrusion_status)
            time.sleep(0.5)
#Логування завершення симуляції виявлення вторгнень
        self.log_event("Система виявлення вторгнень завершила роботу")
#Метод для симуляції оновлення програмного забезпечення пристроїв (firmware)
    def update_device_firmware(self, device_id):
#Перевірка наявності пристрою для оновлення прошивки
        if device_id not in self.devices:
            self.log_event("Оновлення прошивки не вдалося: пристрій " +
device_id + " не знайдено")
            return False
#Симуляція процесу оновлення прошивки з використанням затримок
        self.log_event("Початок оновлення прошивки для пристрою " + device_id)
        for step in range(1, 6):
            self.log_event("Крок " + str(step) + " оновлення прошивки...")
            time.sleep(0.5)
        self.log_event("Прошивку пристрою " + device_id + " оновлено успішно")
        return True
#Метод для симуляції регулярного резервного копіювання даних системи
    def backup_system_data(self):
#Логування початку резервного копіювання даних
        self.log_event("Початок резервного копіювання даних системи")
#Симуляція процесу резервного копіювання з генерацією випадкових даних
        for backup_step in range(1, 4):
            self.log_event("Резервне копіювання, крок " + str(backup_step) + " з
3...")
            time.sleep(0.6)
#Логування завершення процесу резервного копіювання
        self.log_event("Резервне копіювання даних завершено")
#Метод для симуляції оновлення конфігураційних файлів системи
    def update_system_configuration(self):
#Логування початку оновлення конфігурації
        self.log_event("Початок оновлення конфігурації системи")
#Симуляція процесу оновлення конфігурації з декількома кроками
```

```
        steps = ["Оновлення мережевих параметрів", "Оновлення політик безпеки",
"Оновлення доступу користувачів"]
        for step in steps:
            self.log_event("Виконання: " + step)
            time.sleep(0.4)
#Логування завершення оновлення конфігурації системи
        self.log_event("Конфігурацію системи оновлено успішно")

#Функція для генерації хеша з використанням алгоритму SHA256
def generate_hash(input_str):
#Створення об’єкта хешування з вхідного рядка
    hash_object = hashlib.sha256(input_str.encode())
#Повернення шістнадцяткового представлення хеша
    return hash_object.hexdigest()

#Функція для симуляції обробки запитів з різними операціями
def process_user_query(nac_controller):
#Логування початку обробки запитів користувача
    nac_controller.log_event("Початок обробки запитів користувача")
#Багаторазове виконання симуляції запитів для різноманітності операцій
    for query_count in range(2):
        dummy = input("Введіть тестовий запит (натисніть Enter для продовження):
")
        nac_controller.log_event("Оброблено тестовий запит: " + dummy)
        time.sleep(0.3)
#Логування завершення обробки запитів користувача
    nac_controller.log_event("Обробка тестових запитів завершена")

#Головна функція для роботи інтерактивного меню системи Cisco NAC
def main():
#Створення екземпляру контролера Cisco NAC
    nac_controller = CiscoNACController()
#Реєстрація початкових користувачів у системі
    nac_controller.register_user("admin", "adminpass")
    nac_controller.register_user("user1", "user1pass")
    nac_controller.register_user("user2", "user2pass")
#Реєстрація початкових пристроїв у системі
    nac_controller.register_device("DEV001", "AA:BB:CC:DD:EE:01", "192.168.0.2",
"admin")
    nac_controller.register_device("DEV002", "AA:BB:CC:DD:EE:02", "192.168.0.3",
"user1")
    nac_controller.register_device("DEV003", "AA:BB:CC:DD:EE:03", "192.168.0.4",
"user2")
#Додавання початкових політик мережевого доступу
    nac_controller.add_policy("POLICY1", "Дозвіл для пристроїв адміністратора",
["admin"], ["DEV001"])
    nac_controller.add_policy("POLICY2", "Дозвіл для користувачів", ["user1",
"user2"], ["DEV002", "DEV003"])
#Виклик функції обробки тестових запитів
    process_user_query(nac_controller)
#Головний цикл роботи інтерактивного меню
    while True:
        print("\nМеню системи Cisco NAC")
        print("1. Зареєструвати нового користувача")
        print("2. Зареєструвати новий пристрій")
        print("3. Аутентифікувати пристрій")
        print("4. Застосувати політику до пристрою")
        print("5. Провести сканування безпеки пристрою")
        print("6. Оновити IP-адресу пристрою")
        print("7. Заблокувати пристрій")
        print("8. Розблокувати пристрій")
        print("9. Отримати статус пристрою")
        print("10. Запустити діагностику мережі")
        print("11. Запитати деталі пристрою")
        print("12. Симулювати мережевий трафік")
        print("13. Оновити політику мережевого доступу")
        print("14. Видалити політику")
        print("15. Відобразити всі логи")
        print("16. Відобразити всі пристрої")
```

```
        print("17. Відобразити всіх користувачів")
        print("18. Відобразити всі політики")
        print("19. Симулювати аналіз трафіку")
        print("20. Симулювати виявлення вторгнень")
        print("21. Оновити прошивку пристрою")
        print("22. Резервне копіювання даних системи")
        print("23. Оновити конфігурацію системи")
        print("24. Вийти")
        choice = input("Введіть свій вибір (1-24): ")
        if choice == "1":
            username = input("Введіть ім'я нового користувача: ")
            password = input("Введіть пароль для користувача: ")
            nac_controller.register_user(username, password)
        elif choice == "2":
            device_id = input("Введіть ID пристрою: ")
            mac_address = input("Введіть MAC-адресу пристрою: ")
            ip_address = input("Введіть IP-адресу пристрою: ")
            owner_username = input("Введіть ім'я власника пристрою: ")
            nac_controller.register_device(device_id, mac_address, ip_address,
owner_username)
        elif choice == "3":
            device_id = input("Введіть ID пристрою для аутентифікації: ")
            password = input("Введіть пароль власника пристрою: ")
            nac_controller.authenticate_device(device_id, password)
        elif choice == "4":
            device_id = input("Введіть ID пристрою для застосування політики: ")
            policy_id = input("Введіть ID політики: ")
            nac_controller.enforce_policy(device_id, policy_id)
        elif choice == "5":
            device_id = input("Введіть ID пристрою для сканування безпеки: ")
            nac_controller.scan_device_security(device_id)
        elif choice == "6":
            device_id = input("Введіть ID пристрою для оновлення IP: ")
            new_ip = input("Введіть нову IP-адресу: ")
            nac_controller.update_device_ip(device_id, new_ip)
        elif choice == "7":
            device_id = input("Введіть ID пристрою для блокування: ")
            nac_controller.block_device(device_id)
        elif choice == "8":
            device_id = input("Введіть ID пристрою для розблокування: ")
            nac_controller.unblock_device(device_id)
        elif choice == "9":
            device_id = input("Введіть ID пристрою для отримання статусу: ")
            status = nac_controller.get_device_status(device_id)
            print(status)
        elif choice == "10":
            nac_controller.run_diagnostic()
        elif choice == "11":
            device_id = input("Введіть ID пристрою для запиту деталей: ")
            details = nac_controller.query_device(device_id)
            if details is not None:
                for key, value in details.items():
                    print(key + ": " + str(value))
        elif choice == "12":
            nac_controller.simulate_network_traffic()
        elif choice == "13":
            policy_id = input("Введіть ID політики для оновлення: ")
            update_type = input("Оновити дозволених користувачів (u) чи
пристроїв (d)? ")
            if update_type.lower() == "u":
                users_input = input("Введіть імена користувачів через кому: ")
                users_list = [user.strip() for user in users_input.split(",")]
                nac_controller.update_policy(policy_id,
allowed_users=users_list)
            elif update_type.lower() == "d":
                devices_input = input("Введіть ID пристроїв через кому: ")
                devices_list = [device.strip() for device in
devices_input.split(",")]
```

```
                nac_controller.update_policy(policy_id,
allowed_devices=devices_list)
            else:
                print("Невірно обраний тип оновлення")
        elif choice == "14":
            policy_id = input("Введіть ID політики для видалення: ")
            nac_controller.remove_policy(policy_id)
        elif choice == "15":
            nac_controller.display_all_logs()
        elif choice == "16":
            nac_controller.display_all_devices()
        elif choice == "17":
            nac_controller.display_all_users()
        elif choice == "18":
            nac_controller.display_all_policies()
        elif choice == "19":
            nac_controller.simulate_traffic_analysis()
        elif choice == "20":
            nac_controller.simulate_intrusion_detection()
        elif choice == "21":
            device_id = input("Введіть ID пристрою для оновлення прошивки: ")
            nac_controller.update_device_firmware(device_id)
        elif choice == "22":
            nac_controller.backup_system_data()
        elif choice == "23":
            nac_controller.update_system_configuration()
        elif choice == "24":
            print("Вихід із системи Cisco NAC. До побачення!")
            break
        else:
            print("Невірний вибір. Введіть число від 1 до 24.")
        time.sleep(1)
#Завершення роботи програми
    sys.exit(0)

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

Файл DBLogger.py

```
#!/usr/bin/env python3
import tkinter as tk
from tkinter import ttk
import sqlite3
import time
import random
import threading
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import sys

class DBLogger:
    def __init__(self, db_file="events.db"):
        self.conn = sqlite3.connect(db_file, check_same_thread=False)
        self.cursor = self.conn.cursor()
        self.cursor.execute("CREATE TABLE IF NOT EXISTS events (id INTEGER
PRIMARY KEY AUTOINCREMENT, timestamp TEXT, event TEXT)")
        self.conn.commit()
    def log_event(self, event):
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
        self.cursor.execute("INSERT INTO events (timestamp, event) VALUES (?,
?)", (timestamp, event))
        self.conn.commit()
    def get_logs(self):
        self.cursor.execute("SELECT timestamp, event FROM events ORDER BY id
DESC")
        return self.cursor.fetchall()
    def clear_logs(self):
        self.cursor.execute("DELETE FROM events")
        self.conn.commit()

class MLAnalyzer:
    def __init__(self):
        self.model = RandomForestClassifier(n_estimators=10)
        self.X = np.random.rand(100, 5)
        self.y = np.random.choice([0, 1], size=100)
        self.trained = False
    def train_model(self):
        X_train, X_test, y_train, y_test = train_test_split(self.X, self.y,
test_size=0.2)
        self.model.fit(X_train, y_train)
        self.trained = True
    def analyze_traffic(self, features):
        if not self.trained:
            self.train_model()
        features = np.array(features).reshape(1, -1)
        result = self.model.predict(features)
        return "Anomaly" if result[0] == 1 else "Normal"
    def simulate_traffic_data(self):
        return list(np.random.rand(5))
```

```
class Role:
    def __init__(self, name):
        self.name = name
        self.permissions = set()
    def add_permission(self, permission):
        self.permissions.add(permission)
    def remove_permission(self, permission):
        if permission in self.permissions:
            self.permissions.remove(permission)

class Permission:
    def __init__(self, name, description=""):
        self.name = name
        self.description = description

class RBACManager:
    def __init__(self):
        self.roles = {}
        self.user_roles = {}
    def add_role(self, role_name):
        if role_name not in self.roles:
            self.roles[role_name] = Role(role_name)
    def remove_role(self, role_name):
        if role_name in self.roles:
            del self.roles[role_name]
            for user in self.user_roles:
                if role_name in self.user_roles[user]:
                    self.user_roles[user].remove(role_name)
    def assign_role(self, username, role_name):
        if role_name not in self.roles:
            self.add_role(role_name)
        if username not in self.user_roles:
            self.user_roles[username] = set()
        self.user_roles[username].add(role_name)
    def revoke_role(self, username, role_name):
        if username in self.user_roles and role_name in
self.user_roles[username]:
            self.user_roles[username].remove(role_name)
    def add_permission_to_role(self, role_name, permission):
        if role_name in self.roles:
            self.roles[role_name].add_permission(permission)
    def check_permission(self, username, permission):
        if username not in self.user_roles:
            return False
        for role_name in self.user_roles[username]:
            if permission in self.roles[role_name].permissions:
                return True
        return False

class UserActivityAnalytics:
    def __init__(self):
        self.activity_log = []
```

```
    def log_activity(self, username, activity):
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
        self.activity_log.append((timestamp, username, activity))
    def generate_report(self):
        report = {}
        for entry in self.activity_log:
            username = entry[1]
            report[username] = report.get(username, 0) + 1
        sorted_report = sorted(report.items(), key=lambda x: x[1], reverse=True)
        return sorted_report
    def print_detailed_log(self):
        for entry in self.activity_log:
            print(f"{entry[0]} - {entry[1]}: {entry[2]}")

class NetworkMonitorGUI:
    def __init__(self, db_logger, rbac_manager, ml_analyzer,
activity_analytics):
        self.db_logger = db_logger
        self.rbac_manager = rbac_manager
        self.ml_analyzer = ml_analyzer
        self.activity_analytics = activity_analytics
        self.root = tk.Tk()
        self.root.title("Network Monitor")
        self.device_frame = ttk.Frame(self.root)
        self.device_frame.pack(side="top", fill="both", expand=True)
        self.log_frame = ttk.Frame(self.root)
        self.log_frame.pack(side="bottom", fill="both", expand=True)
        self.device_list = tk.Listbox(self.device_frame, height=10)
        self.device_list.pack(side="left", fill="both", expand=True)
        self.log_list = tk.Listbox(self.log_frame, height=10)
        self.log_list.pack(side="left", fill="both", expand=True)
        self.refresh_button = ttk.Button(self.root, text="Refresh",
command=self.refresh)
        self.refresh_button.pack(side="bottom")
        self.populate_devices()
        self.refresh_logs()
        self.root.after(5000, self.auto_refresh)
    def populate_devices(self):
        self.device_list.delete(0, tk.END)
        for i in range(1, 21):
            status =
self.ml_analyzer.analyze_traffic(self.ml_analyzer.simulate_traffic_data())
            self.device_list.insert(tk.END, f"Device {i}: {status}")
    def refresh_logs(self):
        self.log_list.delete(0, tk.END)
        logs = self.db_logger.get_logs()
        for log in logs:
            self.log_list.insert(tk.END, f"{log[0]} - {log[1]}")
    def refresh(self):
        self.populate_devices()
        self.refresh_logs()
    def auto_refresh(self):
        self.refresh()
```

```
        self.root.after(5000, self.auto_refresh)
    def run(self):
        self.root.mainloop()

def simulate_system_operations(db_logger, rbac_manager, ml_analyzer,
activity_analytics):
    users = ["alice", "bob", "charlie", "david", "eve"]
    actions = ["login", "logout", "upload", "download", "update"]
    for i in range(50):
        user = random.choice(users)
        action = random.choice(actions)
        db_logger.log_event(f"{user} performed {action}")
        activity_analytics.log_activity(user, action)
        time.sleep(0.1)
    rbac_manager.assign_role("alice", "admin")
    perm1 = Permission("access_dashboard", "Access to dashboard")
    rbac_manager.add_permission_to_role("admin", perm1.name)
    rbac_manager.assign_role("bob", "user")
    perm2 = Permission("view_data", "View data")
    rbac_manager.add_permission_to_role("user", perm2.name)
    rbac_manager.assign_role("charlie", "moderator")
    perm3 = Permission("moderate_content", "Moderate content")
    rbac_manager.add_permission_to_role("moderator", perm3.name)
    report = activity_analytics.generate_report()
    print("User Activity Report")
    for user, count in report:
        print(f"{user}: {count} actions")
    result = ml_analyzer.analyze_traffic(ml_analyzer.simulate_traffic_data())
    print("Traffic Analysis:", result)
    permission_check = rbac_manager.check_permission("alice",
"access_dashboard")
    print("Alice access_dashboard permission:", permission_check)

def main():
    db_logger = DBLogger()
    rbac_manager = RBACManager()
    ml_analyzer = MLAnalyzer()
    activity_analytics = UserActivityAnalytics()
    threading.Thread(target=simulate_system_operations, args=(db_logger,
rbac_manager, ml_analyzer, activity_analytics)).start()
    gui = NetworkMonitorGUI(db_logger, rbac_manager, ml_analyzer,
activity_analytics)
    gui.run()

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

Файл BackupManager.py

```
import os
import shutil
import time
import threading
import random
import datetime
import smtplib
import socket
import zipfile
import requests

class BackupManager:
    def __init__(self, config_dir='config', backup_dir='backup'):
        self.config_dir = config_dir
        self.backup_dir = backup_dir
        if not os.path.exists(self.backup_dir):
            os.makedirs(self.backup_dir)
    def perform_backup(self):
        timestamp = datetime.datetime.now().strftime('%Y%m%d_%H%M%S')
        backup_filename = os.path.join(self.backup_dir,
f'config_backup_{timestamp}.zip')
        with zipfile.ZipFile(backup_filename, 'w', zipfile.ZIP_DEFLATED) as
backup_zip:
            for foldername, subfolders, filenames in os.walk(self.config_dir):
                for filename in filenames:
                    file_path = os.path.join(foldername, filename)
                    arcname = os.path.relpath(file_path, self.config_dir)
                    backup_zip.write(file_path, arcname)
        return backup_filename
    def schedule_backup(self, interval_seconds=60):
        def backup_loop():
            while True:
                self.perform_backup()
                time.sleep(interval_seconds)
        t = threading.Thread(target=backup_loop)
        t.daemon = True
        t.start()
    def restore_backup(self, backup_filename):
        if os.path.exists(self.config_dir):
            shutil.rmtree(self.config_dir)
        os.makedirs(self.config_dir)
        with zipfile.ZipFile(backup_filename, 'r') as backup_zip:
            backup_zip.extractall(self.config_dir)
    def list_backups(self):
        backups = [f for f in os.listdir(self.backup_dir) if
f.startswith('config_backup_') and f.endswith('.zip')]
        backups.sort()
        return backups

class Notifier:
```

```
    def __init__(self, smtp_server='smtp.example.com', smtp_port=587,
sender_email='noreply@example.com', sender_password='password'):
        self.smtp_server = smtp_server
        self.smtp_port = smtp_port
        self.sender_email = sender_email
        self.sender_password = sender_password
        self.notification_queue = []
    def send_email(self, recipient_email, subject, message):
        email_message = f"Subject: {subject}\n\n{message}"
        try:
            server = smtplib.SMTP(self.smtp_server, self.smtp_port)
            server.starttls()
            server.login(self.sender_email, self.sender_password)
            server.sendmail(self.sender_email, recipient_email, email_message)
            server.quit()
            return True
        except Exception as e:
            return False
    def send_sms(self, phone_number, message):
        time.sleep(0.5)
        return True
    def schedule_notification(self, recipient, subject, message, delay_seconds):
        def delayed_send():
            time.sleep(delay_seconds)
            self.send_email(recipient, subject, message)
        t = threading.Thread(target=delayed_send)
        t.start()
    def aggregate_notifications(self, notifications):
        aggregated = "\n".join(notifications)
        return aggregated

class ThreatDetector:
    def __init__(self):
        self.threats = []
    def detect_threats(self, log_entries):
        detected = []
        for entry in log_entries:
            result = self.analyze_log(entry)
            if result:
                detected.append(result)
        self.threats.extend(detected)
        return detected
    def analyze_log(self, log_entry):
        threat_keywords = ['attack', 'intrusion', 'malware', 'ddos', 'phishing']
        for keyword in threat_keywords:
            if keyword in log_entry.lower():
                return f"Threat detected: {keyword} in log: {log_entry}"
        return None
    def simulate_threat_detection(self):
        simulated_logs = []
        for i in range(10):
            if random.random() < 0.3:
```

```
                simulated_logs.append(f"Suspicious activity detected:
{random.choice(['attack', 'intrusion', 'malware', 'ddos', 'phishing'])}")
            else:
                simulated_logs.append("Normal operation log")
        return self.detect_threats(simulated_logs)
    def get_threat_report(self):
        report = "\n".join(self.threats)
        return report

class SIEMIntegrator:
    def __init__(self, siem_server='siem.example.com', siem_port=514):
        self.siem_server = siem_server
        self.siem_port = siem_port
        self.connection = None
    def check_connection(self):
        try:
            self.connection = socket.create_connection((self.siem_server,
self.siem_port), timeout=5)
            self.connection.close()
            return True
        except Exception as e:
            return False
    def format_logs(self, logs):
        formatted = ""
        for log in logs:
            formatted += f"{log[0]} - {log[1]}\n"
        return formatted
    def export_logs(self, logs):
        formatted_logs = self.format_logs(logs)
        return formatted_logs
    def simulate_sending_logs(self, logs):
        if self.check_connection():
            formatted_logs = self.export_logs(logs)
            time.sleep(1)
            return f"Logs sent to SIEM server:\n{formatted_logs}"
        else:
            return "Connection to SIEM server failed"

class AutoUpdater:
    def __init__(self, current_version='1.0.0',
update_url='https://updates.example.com/latest'):
        self.current_version = current_version
        self.update_url = update_url
        self.available_update = None
    def check_for_updates(self):
        try:
            simulated_response = {'version': '1.1.0', 'download_url':
'https://updates.example.com/download/1.1.0'}
            if simulated_response['version'] != self.current_version:
                self.available_update = simulated_response
                return True
            else:
                return False
```

```
            except Exception as e:
                return False
        def download_update(self):
            if self.available_update:
                time.sleep(2)
                update_file = f"update_{self.available_update['version']}.zip"
                with open(update_file, 'w') as f:
                    f.write("Simulated update content")
                return update_file
            return None
        def install_update(self, update_file):
            time.sleep(1)
            self.current_version = self.available_update['version']
            return True
        def rollback_update(self):
            time.sleep(1)
            self.current_version = '1.0.0'
            return True
        def schedule_update_check(self, interval_seconds=300):
            def update_loop():
                while True:
                    if self.check_for_updates():
                        update_file = self.download_update()
                        installed = self.install_update(update_file)
                        if not installed:
                            self.rollback_update()
                    time.sleep(interval_seconds)
            t = threading.Thread(target=update_loop)
            t.daemon = True
            t.start()
        def verify_update(self):
            if self.available_update and self.current_version ==
    self.available_update['version']:
                return True
            return False
        def simulate_update_process(self):
            if self.check_for_updates():
                update_file = self.download_update()
                success = self.install_update(update_file)
                if success:
                    return f"Updated to version {self.current_version}"
                else:
                    self.rollback_update()
                    return f"Update failed, rolled back to version
    {self.current_version}"
            else:
                return "No updates available"

    def main():
        backup_manager = BackupManager()
        notifier = Notifier()
        threat_detector = ThreatDetector()
        siem_integrator = SIEMIntegrator()
```

```
    auto_updater = AutoUpdater()
    backup_manager.schedule_backup(10)
    notifier.schedule_notification("user@example.com", "Test Notification",
"This is a test", 5)
    threats = threat_detector.simulate_threat_detection()
    print(threat_detector.get_threat_report())
    sample_logs = [("2025-03-05 12:00:00", "User logged in"), ("2025-03-05
12:05:00", "User performed attack")]
    siem_result = siem_integrator.simulate_sending_logs(sample_logs)
    print(siem_result)
    updater_result = auto_updater.simulate_update_process()
    print(updater_result)
    time.sleep(2)
    backups = backup_manager.list_backups()
    print("Available backups:", backups)

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