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 кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу”

Виконав здобувач вищої освіти
IV курсу, групи КБ-22-МБ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Шупляков Н.О.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.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 *125 "*Кібербезпека*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Кібербезпека*"*_

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

Шуплякову Нікіті Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 51-02 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 кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу*

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 кібербезпеки в промислову експлуатацію.
6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*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
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Шупляков Н.О.
(прізвище та ініціали)

АНОТАЦІЯ

Шупляков Н.О. Програмне забезпечення системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

Метою розробки є програмне забезпечення системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

Результат роботи – програмна реалізація системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

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: кібербезпека, зловмисний вплив на мережу

ABSTRACT

Shuplyakov N.O. Software for the cybersecurity system of intelligent services to counteract malicious influence on the network. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the cybersecurity system of intelligent services to counteract malicious influence on the network.

The purpose of the development is the software for the cybersecurity system of intelligent services to counteract malicious influence on the network.

The result of the work is the software implementation of the cybersecurity system of intelligent services to counteract malicious influence on the network.

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 PCs with Windows 10/11.

The program is developed in the Python environment.

Keywords: cybersecurity, malicious influence on the network

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. Багато підприємств дотепер будують свою систему захисту опираючись на вже застарілий периметровий підхід, зосереджуючи всі засоби безпеки в одній-двох контрольних точках мережі, повністю забуваючи про наявність обхідних каналів – Wi-Fi, флешок, 4G, ActiveSync і т.п. Та й про внутрішнього порушника, що вже перебуває усередині мережі й може виконувати своє “чорна справа”, не боячись бути виявленим периметровими засобами захисту, багато хто теж забувають.

Існує варіант – побудувати у внутрішній мережі ще одну, але вже накладену мережу із засобів безпеки? Я думаю будь-який виробник засобів захисту із задоволенням підготує пропозицію по даному варіанті, включивши в нього безліч сенсорів IPS і міжмережних екранів, які будуть моніторити і контролювати внутрішні мережні потоки й виявляти шкідливий код і недозволені додатки. Але такий варіант сполучений з рядом труднощів. По-перше, не завжди існуючий дизайн мережі дозволяє реалізувати таке підключення. то мережа працює на швидкостях, непідвласних засобам захисту, то span-порти для підключення IDS уже зайняті, то підприємство активно задіє віртуалізацію й засобу захисту не можуть ефективно контролювати трафік, що не йде за межі фізичного сервера. По-друге, установка додаткових пристроїв у внутрішній мережі вимагає чималих фінансових засобів, що в умовах непростої економічної ситуації не завжди реалізовано.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

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

– Огляд існуючих систем інтелектуальних сервісів протидії зловмисному впливу на мережу.

– Дослідження системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

– Програмна реалізація системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі інтелектуальних сервісів протидії зловмисному впливу на мережу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу, 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

Існує варіант, що рекомендується часто, – установити засоби захисту на сервера й робочі станції, виявляючи й блоку несанкціоновану активність користувачів або проти користувачів. Всі так, але... Що робити із принтерами, сканерами, промисловими контролерами, IP-системами відеоспостереження й контролю доступу? Адже за ними користувачі не працюють (і їх не можна автентифікувати традиційними методами) і на них не можна поставити ні антивірус, ні HIPS, ні інші засоби захисту. Так уже зложилося, що часто ці пристрої, а їх може бути навіть більше, ніж користувальницьких комп'ютерів, стають мішенню для зловмисників або площадкою для подальшого просування по внутрішній мережі підприємства. На такого роду пристроях можна встановити перехоплювач трафіку й він буде сніффить все, до чого він може дотягтися. І ніякі засоби захисту ПК і серверів такі порушення політики безпеки не помітять у принципі. А наявність обхідних каналів у вигляді незахищеного Wi-Fi або 4G-модеми приведуть до того, що конфіденційні дані можуть витекти минаючи засобу захисту корпоративного периметра.

А може бути спробувати покласти це завдання на те, що й так є й у що інвестовано чималі засоби? Мова йде про мережну інфраструктуру, про маршрутизатори, комутатори й точки доступу, які можуть не тільки передавати трафік із точки А в точку Б, але й ефективно захищати цей трафік, виконуючи одночасно роль сенсора, захисної стіни й інструмента реагування на інциденти безпеки. Адже по суті кожний мережний пристрій являє собою сенсор системи захисту мережі – трафік проходить через нього, трафік ідентифікується й класифікується, трафік направляється в точку призначення. Чому б не зробити ще один крок і не додати до кожного із цих пунктів фразу «з погляду політики ІБ»?

Чому не можна ідентифікувати додатка на рівні маршрутизатора або комутатора, не доводячи їх трафік до MME на периметрі? Чому не можна ідентифікувати атаки, не комутируючи трафік через span-порт на IDS, а скориставшись можливостями інфраструктурного встаткування? Чому не можна блокувати трафік на порту комутатора, до якого підключається порушник, а не чекати, коли трафік дійде до MME? Чому не можна динамічно міняти списки контролю доступу залежно від місця розташування користувача або пристрою, а не закривати око на неконтрольоване ходіння трафіку усередині мережі й необмежений доступ користувачів до внутрішніх ресурсів?

А чому, властиво, не можна? Можна! Саме це й робить Cisco у своїй мережній інфраструктурі, що виступає не тільки як сенсор системи захисту (Network as a Sensor), але і як система захисту (Network as a Enforcer) і система розслідування інцідентів ІБ. У якості вихідних даних ми використовуємо протокол Netflow, що дає нам всі потрібні дані про минаючий трафік, що відповідають на всі важливі для політики ІБ питання – хто, що, коли, куди / звідки, як. За допомогою NetFlow ми можемо класифікувати трафік, розпізнавати додатка, ідентифікувати атаки й витоки, виявляти використання недозволених додатків або поява сторонніх вузлів, проводити розслідування інцідентів і ідентифікувати точку входу зловмисників у мережу. Все це дозволяє зробити Netflow, на який накладається аналітика ІБ, закладена в рішення Cisco Cyber Threat Defense. Розмежування й блокування несанкціонованого доступу реалізується за допомогою списків контролю доступу й міток безпеки SGT (Security Group Tag), що закладають основи для технології Cisco TrustSec, а ефективно управляти всіма налаштуваннями безпеки на десятках тисяч пристроїв допомагає Cisco Identity Service Engine (ISE).

Чим цікаво таке рішення по мережній безпеці, убудоване в саму мережну інфраструктуру? Крім рішення завдань безпеки внутрішньої мережі й локалізації проблем у момент їхнього виникнення, а не тоді, коли вони досягають периметра або деяких контрольних точок у мережі, у яких ставляться засоби захисту, ми

одержуємо й ще одну перевагу – захист зроблених в інфраструктуру інвестицій і можливість забезпечувати захист підприємства в умовах урізування бюджетів. Адже інфраструктура в нас уже є. Усе, що нам потрібно – це система аналітики й візуалізації Netflow з погляду ІБ Cisco Cyber Threat Defense, а також система керування динамічними списками контролю доступу – Cisco Identity Service Engine.

Захист периметра й установку IDS / IPS / MME усередині мережі однаково ніхто не скасовував. Просто сама по собі вона вже не рятує. В ідеалі вона повинна бути інтегрована із захистом внутрішньої мережі й функціонувати по єдиних політиках.

1.2 Область застосування

На своїй щорічній конференції Cisco Live (цього разу пройшла в Сан-Дієго) компанія Cisco представила нові рішення, покликані забезпечити безпека й значно поліпшити можливості моніторингу й контролю на всьому протязі розширеної мережі – від центрів обробки даних, хмарних інфраструктур і віддалених офісів до прикінцевих пристроїв. Інтеграція технологій повсюдної безпеки дозволить замовникам і постачальникам послуг використовувати переваги загрозоорієнтованого захисту, оскільки саме такий тип захисту найбільш актуальний для протидії сучасному динамічному ландшафту погроз. Тільки при забезпеченні повсюдного захисту можна без зайвих ризиків використовувати можливості, надавані цифровою економікою й Всеосяжним Інтернетом (Internet of Everything, IoE).

Як очікується, протягом наступного десятиліття ринок Всеосяжного Інтернету може принести прибуток в 19 трлн доларів США, а можливості, які він створить для постачальників послуг, оцінюються в 1,7 трлн доларів США. Крім того, за прогнозом Cisco, опублікованому в щорічному звіті «Наочний індекс розвитку мережних технологій» (Cisco® Visual Networking Index, Cisco VNI) за

2024 рік, у період з 2024 по 2027 р. кількість міжмашиних з'єднань і персональних пристроїв, підключених до Інтернету, виросте з нинішніх 14 млрд до більш ніж 24 млрд. Разом з тим, однак, росте активність кіберзлочинців, удосконалюються їхні методи й технічна оснащеність. Це обумовлено тим, що фінансові вигоди, одержувані зловмисниками, теж збільшується, і зараз, за різними оцінками, становить від 450 млрд до 1 трлн доларів США.

Повсюдна безпека для організацій і постачальників послуг

Спростити процеси забезпечення інформаційної безпеки в умовах розподілених організацій, удосконалити технології виявлення погроз навіть у самих віддалених ділянках обчислювальних інфраструктур – от чого домагалася компанія Cisco при розробці рішень для захисту всього простору розширеної мережі. Для поліпшення можливостей моніторингу були впроваджені додаткові датчики, для посилення захисту – контрольні точки, для скорочення часу виявлення й часу реагування – система всеосяжного, поліпшеного захисту від погроз. Працюючи спільно, ці засоби ефективно протидіють кібератакам. Завдяки технологіям повсюдної безпеки, рішення компанії Cisco забезпечують масштабований захист, що ефективно протидіє найширшому спектру кіберзагроз протягом усього життєвого циклу атаки.

Розширені можливості інформаційної безпеки для організацій

Cisco представляє наступні відновлення для всього набору мережних рішень:

– У тому, що стосується прикінцевих пристроїв: за допомогою об'єднаних можливостей рішень Cisco AnyConnect ® і Cisco AMP для прикінцевих пристроїв замовники, що застосовують Cisco AnyConnect 4.1 VPN Client, тепер можуть із легкістю використовувати й нарощувати можливості безперервного й ретроспективного захисту прикінцевих пристроїв с VPN від удосконалених погроз.

– В офісах і філія: рішення на базі функцій FirePOWER для маршрутизаторів Cisco з інтегрованими сервісами (Cisco® Integrated Services

Routers – ISR) надають централізовано керовану систему запобігання вторгнень нового покоління (NGIPS) і систему поліпшеного захисту від погроз (Cisco® AMP). Ці рішення орієнтовані на такі обчислювальні інфраструктури, де не завжди можна застосовувати виділені пристрої для забезпечення безпеки.

– Мережа як сенсор і захисний пристрій: Cisco впроваджує численні технології безпеки безпосередньо в мережну інфраструктуру, що дозволяє одержати поліпшені можливості моніторингу для швидкої ідентифікації користувачів і пристроїв, які потенційно можуть бути пов'язані з порушеннями нормальних робочих процесів і навіть погроз для мережі й додатків. Нові можливості містять у собі:

– поліпшену інтеграцію між рішеннями Identity Services Engine (ISE) і Lancope StealthWatch. Тепер співробітники служб безпеки можуть не тільки відзначати окремі IP-Адреси, але й ідентифікувати вектори погроз, застосовуючи контекстної даної системи ISE, у тому числі інформацію про те, яким образом користувачі й пристрої одержують доступ і взаємодіють із мережними ресурсами організації. Ця можливість значно поліпшує контекстуальне виявлення погроз, а їхню прискорену ідентифікацію забезпечує технологія StealthWatch;

– підтримку NetFlow на платформах Cisco UCS®. Переваги концепції «мережа як сенсор» тепер поширилися на фізичні й віртуальні сервери. Це надає замовникам можливості поліпшеного контролю за мережним трафіком, а також спеціальну аналітичну інформацію про погрози для центрів обробки даних.

Завдяки новим убудованим можливостям по забезпеченню повсюдного захисту, обчислювальні мережі Cisco тепер можуть самостійно автоматизувати й застосовувати політики безпеки. Замовники одержали можливість виділити в розширеній мережі підприємства окремі групи додатків і користувачів, задати відповідні політики й визначити, які користувачі мають право працювати з певними додатками і який трафік допустимо в мережі. Крім того, на основі цих рішень можуть бути автоматизовані певні функції безпеки.

– Інтеграція технологій TrustSec + ISE і StealthWatch. Рішення StealthWatch тепер може вносити зміни в сегментацію й тим самим блокувати підозрілі мережні пристрої для швидкої протидії ідентифікованим погрозам. Після цього рішення ISE може при необхідності змінити відповідним чином політики доступу для маршрутизаторів, комутаторів і бездротових LAN-Контролерів Cisco, оснащених технологією TrustSec.

Крім того, Cisco анонсувала ще кілька нововведень:

– Рішення Hoste d Identity Services (забезпечує надійний, цілодобово доступний хмарний сервіс для Cisco Identity Service Engine – платформи керування політиками безпеки, що уніфікує, автоматизує й захищає функції контролю мережного доступу). Новий сервіс упорядковує роботу з мобільними пристроями організації, забезпечуючи рольове, контекстно-орієнтоване застосування ідентифікації користувачів і пристроїв, що мають доступ до роботи в мережі. Крім того, завдяки цьому рішенню прискорюються процеси впровадження, що дуже важливо при масштабуванні бізнесу.

– Рішення pxGrid Ecosystem. Екосистема pxGrid Ecosystem розширилася на одинадцять нових партнерів і включила в себе кілька нових груп технологій, що охоплюють, наприклад, хмарну безпеку й керування продуктивністю мереж і додатків. Рішення pxGrid являє собою розроблену компанією Cisco архітектуру обміну контекстною інформацією, завдяки якій різні платформи, що забезпечують інформаційну безпеку, можуть обмінюватися даними для поліпшення загальної ефективності своєї роботи з виявлення погроз і протидії їм.

Безпека інфраструктури Cisco Evolved Programmable Network для постачальників послуг

Рішення, що поставляються компанією Cisco, для безпеки постачальників послуг розроблені із застосуванням унікального підходу, орієнтованого на професійні вимоги постачальників послуг. Ці рішення формують спеціальну загрозоорієнтовану систему захисту, що динамічно забезпечує безпеку робочих

процесів у міру їхньої появи, а також гнучко розподіляється на фізичні, віртуальні й хмарні інфраструктури.

З огляду на потреби постачальників послуг, яким потрібна відкрита, гнучка й програмувальна інфраструктура, компанія Cisco розширила можливості вдосконаленої загрозоорієнтованого захисту, і тепер вони доступні й для рішення Evolved Programmable Network (EPN). Платформа Cisco EPN являє собою надійна основа, що базується на відкритій мережній архітектурі й спроектована для того, щоб використовувати всі переваги технологій програмно-визначаємих мереж (Software Defined Networking, SDN) і віртуалізації мережних функцій (Network Functions Virtualization, NFV). Cisco EPN дозволяє скоротити час окупності, зменшити витрати й технічні складності, пов'язані із впровадженням нових сервісів.

Нові рішення компанії Cisco для безпеки постачальників послуг включають наступні пропозиції:

– Інтегрована платформа Cisco Firepower™ 9300 являє собою високопродуктивне, масштабоване, модульне, багатофункціональне рішення операторського класу, спроектоване спеціально для постачальників послуг. Ця платформа може забезпечувати безпека більших потоків даних, необхідних для форсованої роботи сервісів, і повністю відповідає вимогам, пропонованим до встаткування операторського класу.

– Розширені можливості поліпшеної оркестрації й хмарних технологій уможливлюють легку інтеграцію нових рішень Cisco для інформаційної безпеки з архітектурою Cisco, зі сторонніми SDN / NFV рішеннями, а також з Cisco’s Adaptive Security Appliance Virtual (ASAv), Cisco’s Network Service Orchestrator (NSO) і Application-Centric Infrastructure (ACI). Ці можливості оркестрації й хмарних технологій також містять у собі інтерфейси API для інтеграції із системами підтримки операцій і системами підтримки бізнесу (Operation Support Systems / Business Support Systems), а також хмарними рішеннями виду «безпека як послуга».

– Удосконалені можливості – наприклад, захищені контейнери – забезпечують розміщення сервісів безпеки й додатків, планованих у перспективі. Додатково впроваджена підтримка MME серії Cisco ASA і стороннього рішення від компанії Radware для протидії DDo-Атакам. Крім того, на другу половину 2017 року запланована реалізація ряду додаткових можливостей.

Для того, щоб організація могла ефективно використовувати всі нові можливості й технології для свого росту й розвитку, необхідний надійний захист від сучасних кіберзагроз і підвищена мобільність бізнес-процесів. Це можливо тільки в тому випадку, якщо ефективні засоби інформаційної безпеки будуть впроваджені повсюдно на всьому протязі мережної інфраструктури. Завдяки інтеграції технологій повсюдної безпеки на всьому просторі розширеної мережі й хмарних сервісів, компанія Cisco надійно захищає замовників від широкого спектра погроз. Не менш важлива й стовідсоткова впевненість організацій і постачальників послуг у тому, що вони мають у своєму розпорядженні всі можливості безперервного й ретроспективного моніторингу й контролю для того, щоб забезпечити безпеку всіх бізнес-процесів, пов'язаних із Всеосяжним Інтернетом і цифровою економікою.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу, 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

Почну свій огляд з інформації про зараження комп'ютера, наслідках цього й способах захисту. Відразу хочу відзначити, що на рахунок антивірусних продуктів – скільки людей, стільки й думок.

Зараження комп'ютера вірусами, троянами, шпигунами і як із цім боротися

Як відомо, заражений вірусами комп'ютер, як правило, починає працювати не стабільно й симптоми, а також наслідку зараження можуть бути зовсім різними:

– Падіння продуктивності комп'ютера.

– Несподівані перезавантаження й вимикання комп'ютера.

– Поява незвичайних неконтрольованих дій системи під час звичайної вашої роботи за комп'ютером:

– мимовільне відкриття або закриття папок / файлів / програм;

– зникнення файлів / папок / ярликів;

– неможливість запуску якихось файлів і програм, які завжди запускалися без проблем;

– поява різних баннерів (у тому числі – здирників) на робочому столі або під час завантаження ПК;

– відключення доступу до мережі Інтернет.

І ще багато інших різних несанкціонованих дій з боку системи.

– Втрата інформації, що зберігається на комп'ютері.

– Повна непрацездатність системи (відсутність можливості завантажити операційну систему в наслідок зараження комп'ютера).

– Крадіжка інформації з комп'ютера (логінів, паролів, інших важливих даних).

Захиститися від вірусних і шпигунських атак допомагають усім відомі програми – Антивіруси.

Що стосується платних продуктів, то із всіх і не мають повнофункціональних безкоштовних версій, виділю:

Дуже популярний, рекомендований мною – Антивірус Касперського (Kaspersky Anti-Virus). Незважаючи на те, що він платний, завжди їсти спосіб не купувати антивірус і зробити його безкоштовним різними простими способами (зрозуміла справа – нелегальними) і ними користуються переважну більшість. Серед всіх платних продуктів я сміло виділяю саме його, ґрунтуючись чисто на своєму досвіді від 4-х років. З його використанням, система не піддавалася зараженню, завжди працює стабільно, ніяких навіть дрібних симптомів присутності зараження жодного разу не було виявлено за такий тривалий строк. Звичайно, потрібно враховувати те, що я ніколи не забуваю про основні запобіжні заходи (про які мова йтиме трохи пізніше). У цілому антивірус на 100% справляється зі своїми функціями, але не буду заперечувати його пристойні вимоги до ресурсів комп'ютера.

Інші платні продукти (наприклад: DrWeb, NOD32, Norton) я не розглядаю по одній простій причині: за то час поки я пробував їх застосовувати, вони завжди по різних причинах (захист, інтерфейс, зручність) уступали Касперському. Особливо з головної причини – виявленню вірусної активності й лікуванню вже зараженого комп'ютера.

Безкоштовні повнофункціональні антивірусні продукти:

– Антивірус Avast! Free. Саме його я рекомендую, у випадку якщо купувати антивірус Касперського немає можливості, а прибігати до способів зробити його «безкоштовним» не хочеться з якоїсь причини або можливості

немає. Повністю безкоштовний продукт Avast! Free являє собою повноцінний антивірус із більшою кількістю різних компонентів для захисту вашого комп'ютера від вірусів, шпигунів, троянов. Я використову його для захисту всіх ПК, де працюють користувачі, у яких немає можливості регулярно (приблизно щомісяця) знову активувати того ж Касперського, щоб продовжити ліцензію. Він, на мій погляд, уступає Касперському у своєму захисті від погроз, оскільки були випадки, коли я бачив, що він пропускає погрозу, що Касперський виявляє й усуває. Але такі випадки дуже й дуже рідкі й у цілому з безкоштовних антивірусних продуктів кращого рішення я не бачив. З недоліків – періодично, що з'являються вікна, з рекламою від розроблювачів даного продукту.

– Comodo Antivirus. Із плюсів – є повноцінний антивірусний захист + Firewall (про нього нижче) безкоштовно. З мінусів – вимагає більше ресурсів системи чим, наприклад, Avast! Free і часом може пропускати деякі погрози.

– Avira Free Antivirus. Із плюсів – дуже невисокі вимоги до системних ресурсів і як наслідок антивірус не загальмовує навіть малопродуктивні й середні комп'ютери. Також добре відловлює вірусні погрози, відомі по базах. З мінусів – має середню реакцію на нові невідомі модифікації вірусів, а також вискакують докучливі рекламні вікна від розроблювачів даного продукту.

На цьому короткий огляд про захист комп'ютера від різних вірусних погроз, завершую. І тепер переходимо до захисту від мережних атак – а саме Файєрволах (Firewall) / брандмауерах (означає теж саме).

Захист комп'ютера від несанкціонованого доступу й обмеження доступу програм в Інтернет

Кожна програма, установлена на комп'ютері може мати «Лазівки» / «Діри», через які зловмисник, запустивши шпигуна, може проникнути у ваш комп'ютер, викрасти якісь коштовні дані, паролі, дані рахунків. Для захисту від подібних погроз застосовуються так звані програми – брандмауери / файєрволи (Firewall) / мережні екрани (називаються по-різному), які здатні контролювати вхідний і вихідний трафік комп'ютера.

Я пробував платні й безкоштовні варіанти подібних рішень і саме серед даних продуктів, безкоштовні рішення нічим не уступають платним, а в якихось випадках зовсім перевершують їх.

Нижче опишу два, на мій погляд, найбільш ефективні рішення для захисту комп'ютера від несанкціонованого мережного доступу:

– Comodo Firewall. Як пам'ятаєте, вище я згадував про безкоштовний антивірус від даного виробника. Незважаючи на те, що антивірус має деякі недоліки, Файєрвол від Comodo – продукт дуже високої якості й також є безкоштовним. По незалежних тестах він займав не раз 1-е місця. Якщо ви вирішили користуватися антивірусом від Comodo, то тоді щоб не ставити окремо Файєрвол, найпростіше скористатися рішенням Comodo Internet Security. Воно включає відразу антивірус + файєрвол і теж є безкоштовним.

І нарешті, якщо ви вибрали інший безкоштовний антивірусний продукт, то найкраще в якості файєрволу окремо встановити безкоштовний Comodo Firewall.

Додаткові заходи по захисту вашого комп'ютера

Установивши надійний антивірус, а також у доповненні до нього гарний мережний екран (файєрвол), ви вже дуже надійно захистите свій комп'ютер від усіляких погроз і впливів з Інтернету. Але не варто забувати або ігнорувати стандартні рекомендації, що ставляться до безпеки вашої системи.

Перелічу головні з них:

– Обов'язкове включення відновлень вашої операційної системи Windows. Регулярні відновлення системи дозволяють закрити багато хто «діри» у безпеці й у цілому стабілізувати роботу системи. У мене відновлення настроєні на автоматичне завантаження й установку й перевіряються щодня.

– Обов'язкове автоматичне щоденне відновлення вірусних баз вашого встановленого антивірусу. Працюючи в системі, у якій стоїть антивірус зі старими базами, ви піддаєте її ризику зараження новими погрозами, які з'являються постійно – щодня.

– Намагатися уникати відвідувань різних сайтів порнографічного змісту або сайтів з вірусними погрозами (про це, при їхньому відвідуванні, вас попередить ваш правильно настроєний антивірус). Переглядаючи подібні сайти, ви дуже піддаєте свій комп'ютер ризику зараження.

– Регулярно необхідно робити резервні копії всіх ваших даних, що зберігаються на комп'ютері. Процедура проста й навіть не зажадає багато ваших дій, якщо настроїти цей процес на автоматичне виконання за розкладом.

І варто згадати самий надійний спосіб захисту від вірусів, а також мережних атак – використання однієї з операційних систем сімейства Linux, замість звичної більшості Windows. Але такий спосіб більшості користувачів не під силу у зв'язку з поганою взаємодією з комп'ютером у цілому або з небажанням навчатися роботі в Linux. Але варто пам'ятати, що операційні системи сімейства Linux безпечніше в багато разів, у порівнянні з Windows.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність. Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до

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

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

– у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;

– гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

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

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

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

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

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

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

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

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

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

6. Тестування (автоматизація цього процесу).

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

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

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

Серед найпопулярніших фреймворків для Python:

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

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

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

Популярні Python IDE

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

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 кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

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 кібербезпеки 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 кібербезпеки 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

Архітектура Cisco TrustSec – це система керування безпекою мережі за допомогою міток безпеки Secure Group Tag (SGT), які по своєму потенціалі несуть якщо не революційний (хоча на мій погляд саме такий), то вже точно набагато більше глибокий і просунутий підхід до формування політик доступу в мережу з можливістю їхньої деталізації й застосування прозоро через всю мережу.

Ключовим елементом даної системи є сервер політик, а саме система контролю й обліку доступу в мережу – Cisco Identity Services Engine (ISE), тому говорити про архітектуру TrustSec з відривом від ISE, на мій погляд, було б не правильно.

Окремо хочеться відзначити, що наведені приклади конфігурації не претендують на звання еталонних або рекомендованих у продуктивному середовищі. Конфігурація стенда проходила в лабораторії, де проводилося багато різних тестів і випробувань системи, тому оптимальність наведених налаштувань може бути далека від ідеалу. Проте, всі налаштування абсолютно робітники і їх можна застосовувати для побудови своїх стендів, а головне самонавчання.

Складності класичних моделей керування доступом у мережу.

У класичних варіантах авторизації доступу в мережу традиційно домінували два методи:

– динамічне призначення VLAN на порту комутатора й контролері БЛОМ;

– завантаження динамічного ACL на порт комутатора / контролер бездротової мережі.

При авторизації користувача на пристрої доступу комутатори / контролері БЛОМ / VPN шлюзі була відсутня можливість прив'язки поточної сесії користувача (як сутності, що володіє безліччю параметрів, а не тільки IP) до правил ACL у міжмережному екрані (ММЕ), контентних шлюзах усередині мережі, ЦОД, периметральних пристроях ІБ. Таким чином, конфігурація ММЕ була вкрай статична й фактично авторизація груп користувачів зводилася до статичної прив'язки IP підмереж до логічних груп користувачів і подальшої фільтрації в правилах ACL на ММЕ.

У більше просунутих варіантах різних вендорів, представлених на ринку рішень ІБ, розглядалися можливості інтеграції їх ММЕ й шлюзів контентної фільтрації з AD за допомогою установки різного роду агентів на прикінцеві хости й / або підстроювання доменних контролерів MS AD на можливість експорту інформації про активних користувачів. При таких підходах виникали наступні проблеми:

– Найчастіше для моніторингу активності логінів використовувався Security Log доменного контролера, що не давало різниці в тому залогінен користувач за допомогою термінальної сесії RDP або локально. Не відзначалися події типу Logout, що приводило до виникнення реального ризику при підміні адреси робочої станції у випадку такої “висячої сесії”.

– Було потрібно вносити зміни до реєстру й налаштувань доменних контролерів, одержання для агента облікового запису адміністратора домену, або розширених прав доступу.

– Гостьові користувачі жодним чином такою системою обліку користувачів відслідковуватися не могли.

– Установка агентів на прикінцеві комп'ютери спричиняє додаткове адміністративне навантаження на розгортання таких інсталяцій і їхню підтримку.

Коробкові рішення такого роду не мали інтеграції з NAD (прикінцевими мережними пристроями доступу), що приводило до відсутності архітектурного підходу як такого й гнучкості в інтеграції таких рішень.

Користувач найчастіше був жорстко прив'язаний до свого робочого місця, блоку IP адрес, активним сесіям автентифікації в додатках і на MME, які доводилося багаторазово проходити для одержання доступу в той або інший сегмент мережі. Крім усього іншого на MME ЦОД (наприклад) було неможливо довідатися який був результат проходження процедури оцінки стану користувальницьким пристроєм, тип пристрою користувача й так далі. Передача знань про користувача в систему WEB контентної фільтрації (корпоративного проксі) також була пов'язана із труднощами інтеграції своїх власних агентів, або з необхідністю в організації активної автентифікації через портал.

З не користувальницькими пристроями начебто принтерів / телефонів / відеокамер ситуація обстояла зовсім важко – статичні дозволи на портах їхнього підключення відкривали потенційні діри на рівні доступу в мережу, також пристрій був легко підмінити.

При роботі через термінальні сервера орієнтуватися просто на IP адреса для авторизації не інформативно. Навіть якщо можна видати унікальна адреса для термінальної сесії користувача, MME нічого не знає про конкретного користувача і його додаткові атрибути (начебто тих, що призначаються в AD, можливо про членство в групі й т.д.).

Опис архітектури Cisco TrustSec

Архітектура Cisco TrustSec разом із системою контролю й обліку доступу в мережу Cisco ISE покликані реалізувати гнучкі централізовані сервіси керування користувальницьким і не користувальницьким доступом у мережу й забезпечити:

– Автентифікацію користувальницьких і не користувальницьких пристроїв у мережі по різних методах і протоколам автентифікації з використанням широкого набору інтегрувальних зовнішніх сервісів і баз автентифікації.

– Авторизацію доступу в мережу залежно від:

– Різного набору критеріїв, яким задовольняє сесія. Гнучкість формування критеріїв авторизації забезпечується булевою логікою при написанні правил. Деякі із критеріїв:

– Членство користувача в групі AD / LDAP.

– Параметри автентифікації – атрибути як користувача в сторонній базі автентифікації, полючи сертифіката, тип автентифікації dot1x / mab і тд.

– Місце розташування користувача, що підключається.

– Тип підключення користувача Wired / Wireless / VPN.

– Тип пристрою, що підключається.

– Час підключення.

– Оцінка стану пристрою – NAC Posture.

– Профілювання типів пристроїв, що підключаються, з метою більше гнучкого формування правил авторизації доступу з урахуванням типу пристрою, що підключається, у тому числі цей функціонал допомагає захиститися від підміни пристрою.

– Оцінка стану пристроїв, що підключаються, на предмет відповідності політиці безпеки, проведення таких оцінок як:

– Windows Update: Версії ОС, наявність Service Pack, Hotfix, версія браузера.

– Антивірус установлений / чи свіжої бази.

– Антишпигунське ПЗ встановлено / чи свіжої бази.

– Файлові параметри.

– Сервіси.

– Додатка.

– І багато яких інших параметрів сесії, доступні в убудованій бібліотеці параметрів Cisco ISE.

– Повний цикл керування гостьовим доступом з поділом на ролі, як гостей, так і тих хто цих гостей може створювати.

– Інтеграцію зі сторонніми системами: MDM, СКУД, SourceFire FirePower, Cisco Threat Defense, SIEM системами й іншими партнерськими рішеннями, що підтримують взаємодія з ISE через API, або pxGrid (Cisco Platform Exchange Grid).

– Застосування методів авторизація сесії за результатом перевірки збігу виконуваних умов:

– URL перенапрямок.

– VLAN динамічне призначення.

– DACL завантаження динамічного ACL.

– SGACL динамічне завантаження Secure Group ACL(SGACL) на комутатори, що входять у довірену мережу TrustSec.

– Авторизація VPN доступу.

– SGT – призначення мітки Secure Group Tag.

– Призначення довільних Radius-Атрибутів для авторизації сесії.

– Change of Authorization – зміна авторизації вже встановленої сесії, наприклад при вимиканні політики NAC на прикінцевом пристрої (Приклад: виключили антивірус.)

Важливо розуміти, що всі перераховані вище методи авторизації можуть застосовуватися як роздільно, так і спільно. Метод Radius CoA застосовується в автоматичному режимі:

– при зміні профілю пристрою;

– за результатом чергової перевірки відповідності прикінцевого пристроїв;

– за сигналом від AnyConnect / NAC агента.

Архітектура TrustSec – це насамперед можливість керування доступом у мережі за допомогою груп безпеки Secure Group Tag (SGT). Мітка SGT несе в собі контекст інформації про конкретну сесію доступу в мережу, її параметрах, таких як перераховані нижче:

– час доступу;

– тип доступу в мережу;

– метод автентифікації;

– тип пристрою;

– параметри автентифікації;

– користувач і його параметри;

– оцінка стану пристрою й т.д.

Дана мітка призначається на пристрій прикінцевого доступу сервером контролю й обліку доступу Cisco ISE, що виконує роль RADIUS сервера із сильно розширеними можливостями. Мітка призначається як результат авторизації сесії прикінцевого пристрою, що підключилося, унікально його ідентифікує й переноситься через всю мережу разом із трафіком пристрою. Відбувається так зване тегованні трафіку прикінцевого пристрою. Тегування дає можливість кожному пристрою MME / Маршрутизатору / Комутатору на шляху проходження пакета бачити, ухвалювати рішення щодо фільтрації й логувати транзакції, опираючись не на абсолютно не інформативна IP адреса (якуможна підмінити), а на багатий набір інформації про джерело трафіку.

Що ж із себе представляє мітка безпеки SGT і як вона передається по мережі? Мітка безпеки це число від 1 до 65535, що відповідно займає при тегуванні поле розміром 16 біт у заголовку пакета даних. Мітка включається в поле Cisco Meta Data, що разом з міткою додає в L2 фрейм додатково 20 байт.

Мітки по мережі можуть передаватися двома шляхами:

– Прямим тегуванням пакетів.

– Через службовий протокол обміну мітками – SGT Exchange Protocol over TCP (SXP).

Розглянемо обидва методи окремо.

Методи передачі міток SGT по мережі.

Для того щоб забезпечити тегованні й фактично зміна фрейму на лету без шкоди продуктивності й забезпечення лінійної швидкості передачі даних, пристрій доступу, наприклад комутатор, повинне мати реалізацію тегування на

апаратному рівні, тобто в ASIC (апаратний чипсет). Слід зазначити, що ASIC , що буде робити модифікацію фреймів, стоїть певних грошей і став впроваджуватися в пристрої Cisco відносно недавно. Таким чином, тегованні пакетів підтримує відносно невелика кількість пристроїв, хоча їхній список постійно поповнюється.

З актуальним списком моделей пристроїв, підтримуваних архітектурою Cisco TrustSec можна ознайомитися по посиланню.

На момент написання статті можна говорити про те, що апаратне тегування підтримують всі сучасні комутатори починаючи з 3560x / 3750x, 3650 / 3850 і далі Cat4k, Cat6k, Nexus5k, Nexus7k, ISRG2, ASR, WLC 5760 і інші, деталі по посиланню вище.

В апаратному тегуванні є як плюси, так і певні мінуси. Що стосується плюсів – економія ресурсів мережного встаткування, оскільки не треба використовувати Control-Plane протоколи для зберігання прив'язок IP<->SGT в оперативній пам'яті TCAM, більша масштабованість. Також із плюсів – широкі можливості по захисту від підміни міток, про це окремо нижче. Відносно мінусів:

– Далеко не всім так повезло мати комутатор Catalyst 3 k-X на рівні доступу.

– У випадку, якщо тегованний трафік попадає на пристрій не Cisco (не розуміючі теги), фрейм просто відкидається, що як не задовольняє IEEE стандарту. Мало того, що не багато хто розташовують кампусом повністю побудованому на Cisco, так ще й передача тегів через WAN мережі для багатьох питання досить актуальний через наявність філій і віддалених площадок.

Щоб уникнути перерахованих незручностей і необхідності зробити впровадження архітектури TrustSec гнучко й на більшій кількості платформ, був розроблений протокол обміну мітками SGT між мережними пристроями.

Протокол SXP працює поверх транспортного протоколу TCP на порту 64999.

Сесії SXP установлюються в топології точка-точка між пристроями знаннями, що обмінюються, про мітки. Клієнтська сесія в процесі авторизації

одержує мітку безпеки від Radius сервера (Cisco ISE), дана мітка зберігається в базі прикінцевого пристрою доступу (NAD) разом із прив'язкою поточного IP адреси клієнта. Даною таблицею відповідності IP<->SGT і обмінюються пристрої по протоколі SXP. SXP сесія може будуватися між різними L3 сегментами й хоч через WAN канал до найближчого пристрою, що або може відфільтрувати трафік, ґрунтуючись на мітках, або зможе апаратно затегувати і відправити далі вже тегований трафік.

Відносно ресурсів ЦОД і зокрема серверів, які автентифікувати у мережі по 802.1x ніхто не буде (це й не потрібно), є кілька гнучких рішень для присвоєння їм міток як статично, так і динамічно. Статично можна задавати прив'язку мітки на IP / Інтерфейс комутатора / VLAN. Залежно від конкретної використовуваної моделі комутатора ЦОД можуть бути доступні різні моделі статичного присвоєння мітки. Статичне присвоєння мітки може бути корисно у випадку неможливості реалізації динамічного методу присвоєння міток, наприклад, окремо вартого сервера без віртуалізації, Z-системи, P-системи й інші специфічні архітектури, на яких не можна запустити гіпервізор VMWare ESX.

Динамічний метод – це використання розподіленого віртуального комутатора гіпервізора VMWare ESX від компанії Cisco – Nexus 1000v, що починаючи з версії ПЗ 5.2(1)SV3(1.1) підтримує протокол SXP. Якщо коротко, то цей віртуальний комутатор (має безкоштовну версію) ставиться замість Distributed комутатора (VDS) в ESX ферму. Працює Nexus 1000v як повноцінний модульний комутатор з консоллю й багатим функціоналом. Сервера ESX є його лінійними картами, а супервізори резервируемо запускаються на окремих віртуальних машинах.

На комутаторі Nexus 1000v створюються портові профілі, які надалі будуть привласнені віртуальним машинам. У портовому профілі відбиваються мережні налаштування, які повинні бути успадковані віртуальною машиною

3.2 Розробка структурної схеми

Сьогодні бізнес переживає революцію глобального масштабу. Ноутбуки з підтримкою технології Wi-Fi одержують все більше поширення, і ця обставина сприяє популяризації бездротових корпоративних мереж (WLAN).На відміну від попередніх технологічних стрибків, рушійною силою яких були експерти й професіонали, своїм бурхливим розвитком корпоративні мережі WLAN зобов'язані мобільним користувачам, керівникам компаній, яким з обов'язку служби часто доводиться бувати у відрядженнях, бездротовим додаткам і сучасним сервісам, таким як VoIP поверх Wi-Fi. Бурхливе поширення корпоративних технологій WLAN радикальним образом міняє стиль бізнес-операцій, схеми організації доступу до мережі й центрів обробки даних, а також методи централізованого керування інфраструктурою IT. Можливість підключитися до мережі в будь-який час і в будь-якому місці стає одним із ключових вимог сучасного бізнесу. Мобільність змінює способи ведення бізнесу організаціями. Взаємодія в реальному часі, миттєвий обмін повідомленнями, текстовий пейджинг, голосові сервіси, доступ до мережі під час поїздок і в реальному часі в умовах офісу трансформують бізнес-середовище. В умовах постійно зростаючої конкуренції компанії прагнуть знайти швидкі відповіді на виклики часу й домогтися миттєвих результатів. Сьогодні мережі WLAN здобувають винятково важливе значення для бізнесу. Кінцеві користувачі відчувають волю й гнучкість, які дає їм бездротовий зв'язок, а керівники вищої ланки поступово усвідомлюють конкурентні переваги критично важливих для бізнесу мобільних додатків. Організації впроваджують мережі WLAN, щоб підвищити продуктивність співробітників, створити більше сприятливі умови для спільної роботи й забезпечити більше оперативне реагування на звертання клієнтів.

Зрослі потреби в повсюдному підключенні до мережі ставлять нові завдання перед фахівцями в області мережних технологій, які повинні знайти

способи задовольнити зростаючий попит на мережі WLAN в епоху скромних бюджетів і вбогих ресурсів. Фахівцям стає ясно, що під час відсутності санкціонованої корпоративної бездротової мережі співробітники користуються власними неавторизованими точками доступу, що ставить під погрозу безпека всієї мережі в цілому.

Перед мережними адміністраторами стоїть завдання захистити корпоративну мережу й забезпечити захищений доступ до мереж WLAN для співробітників своєї організації. Їм необхідна бездротова інфраструктура, що має унікальні атрибути радіочастотної (РЧ) технології й підтримує сучасні бізнес-додатки. Адміністратори повинні забезпечити безпека бездротової мережі й одночасно закласти основу для плавної інтеграції нових додатків, що підтримують бездротову технологію. Мережним адміністраторам необхідне рішення WLAN, що використовує всі переваги вже наявних інструментів, знань і мережних ресурсів для максимально економічного рішення питань, пов'язаних із забезпеченням безпеки WLAN, її розгортання й керування.

Cisco Unified Wireless Network – це єдине в галузі рішення, що сполучить у собі провідні й бездротові компоненти, що дозволяє підприємствам без зайвих витрат вирішувати питання, пов'язані із забезпеченням безпеки WLAN, її розгортання й керування. Поєднуючи в собі краще із провідних і бездротових мереж це потужне рішення дозволяє створювати масштабовані, керовані й захищені мережі WLAN з низькою сукупною вартістю володіння. У ньому реалізовані інноваційні радіочастотні можливості, які дозволяють забезпечити доступ у реальному часі до ключового бізнес-додаткам і організувати захищені підключення до мережі корпоративного класу. Рішення Cisco Unified Wireless Network дозволяє забезпечити для бездротових мереж той же рівень захищеності, масштабованості, надійності, зручності експлуатації й керування, до якого організації звикли в процесі експлуатації діючих провідних мереж.

Cisco Unified Wireless Network являє собою інтегроване, всеосяжне рішення, що охоплює всі рівні WLAN, починаючи від клієнтських пристроїв і

точок доступу й закінчуючи мережною інфраструктурою, інструментами мережного адміністрування, засобами інтеграції сучасних бездротових сервісів і прекрасно зарекомендували себе механізмом всесвітньої підтримки, що діє 24 години на добу. Це рішення забезпечує кращий у галузі рівень безпеки бездротової мережі, інновацій і захисту інвестицій. Це єдине рішення, що сполучить у собі інноваційну технологію організації точок доступу із чудовою системою централізованого адміністрування, засобами інтелектуального керування, сервісами виявлення в реальному часі й підтримкою широкого спектра клієнтських пристроїв, сумісних з устаткуванням Cisco.

За рахунок спрощенню процесів розгортання й експлуатації мережі й керування нею Cisco Unified Wireless Network дозволяє знизити сукупні операційні витрати. Завдяки цьому рішенню можна легко управляти десятками, сотнями й тисячами локальних або віддалених точок доступу за допомогою централізованої керуючої консолі. Гнучкість рішення Cisco Unified Wireless Network дозволяє мережним адміністраторам проектувати мережі, що відповідають індивідуальним потребам компанії, будь то мережі з високим рівнем інтеграції або прості оверлейні мережі.

Побудова бездротових мереж корпоративного класу

Рішення Cisco Unified Wireless Network можна використовувати в корпоративних офісах, лікарнях, підприємствах роздрібної торгівлі, виробничих приміщеннях, на складах, в освітніх і фінансових установах, органах місцевої й федеральної влади, а також в інших офісах по усьому світі. Дане рішення підтримує бізнес-додатка всілякого профілю, що використовують функції Wi-Fi, такі як екстрена медична, інвентарний облік, точки роздрібних продажів, системи відеоспостереження, доступ до даних у реальному часі, облік ресурсів і забезпечення видимості мережі.

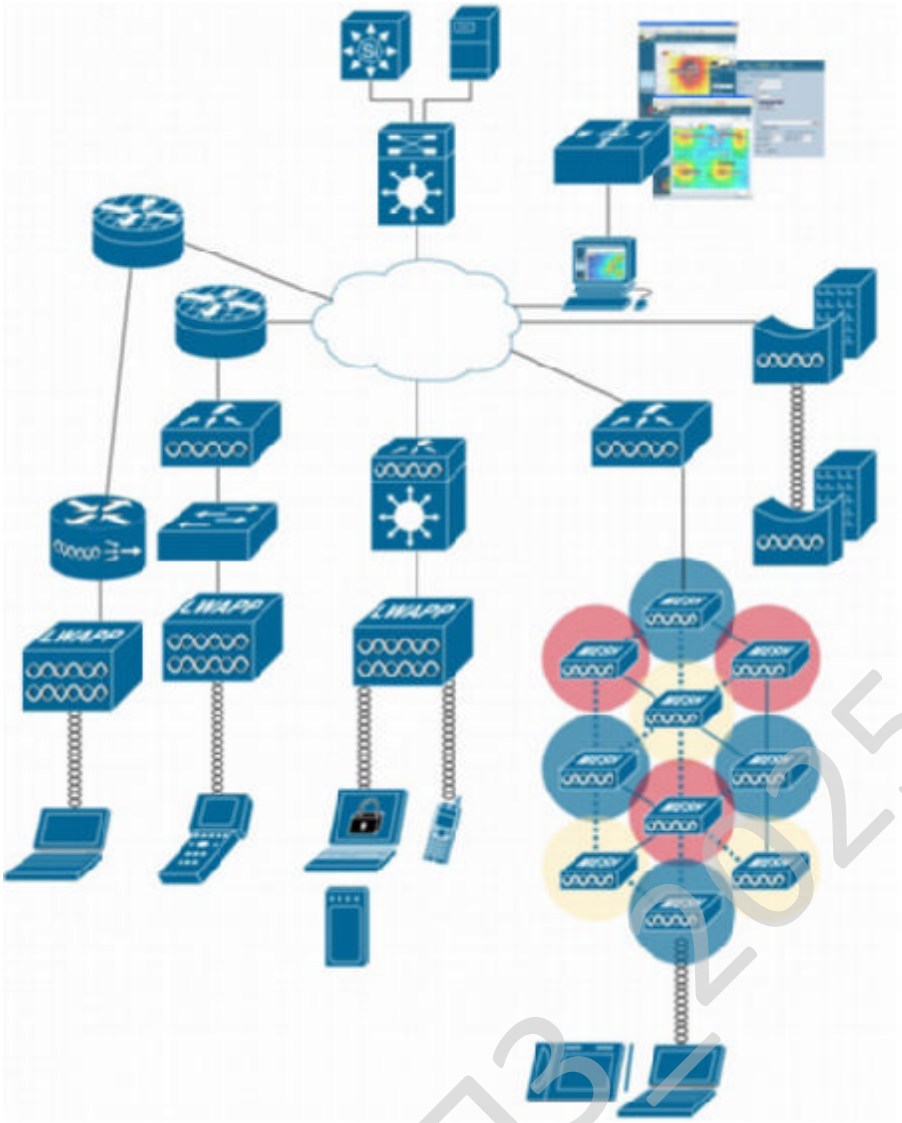
Мобільним користувачам і керівникам різної ланки, що перебуває у відрядженнях, Cisco Unified Wireless Network пропонує можливість доступу до бездротової мережі з таких місць, як публічні хот-споти, готелі, конференц-

центри й аеропорти. Це рішення надає доступ у реальному часі до цілого ряду бізнес-середовищ, забезпечуючи захищену мобільність і гостьовий доступ для кампусів і філій. Замовники можуть сміло впроваджувати рішення Cisco Unified Wireless Network, будучи впевненими в тому, що їхні інвестиції захищені.

Елементи рішення Cisco Unified Wireless Network

Рішення Cisco Unified Wireless Network містить у собі п'ять взаємозалежних елементів, які в сукупності становлять уніфіковане бездротове рішення корпоративного класу. До числа названих елементів ставляться клієнтські пристрої, мобільна платформа, уніфікація мережі, засоби керування мережею світового рівня й уніфіковані додаткові сервіси. Починаючи з базових клієнтських пристроїв, кожний елемент вносить свою лепту в удосконалювання мережі в міру її розвитку й росту, вступаючи у взаємодію з елементами, розташованими на нижньому й верхньому рівнях. У сполученні ці елементи дозволяють створити всеосяжне захищене рішення WLAN. Cisco пропонує широкий спектр продуктів WLAN, що підтримують п'ять взаємодіючих елементів рішення Cisco Unified Wireless Network.

Cisco Unified Wireless Network – це передове, всеосяжне рішення, що охоплює клієнтські пристрої, точки доступу, контролери, комутатори й маршрутизатори, засоби адміністрування світового рівня й додаткові сервіси з підтримкою, що відповідає корпоративним стандартам. Завдяки великим асортиментам продуктів, уніфікованій архітектурі, плавній моделі переходу до майбутніх удосконалень і розширених програм переходу на нові технології пропоноване рішення забезпечує надійний захист інвестицій. Рішення Cisco Unified Wireless Network підтримує критично важливі для бізнесу додатка, що працюють у режимі реального часу, і дозволяє створити захищений, мобільний і інтерактивний робочий простір для організацій, що впроваджують мережі WLAN.



Додаткові сервіси
Голос через Wi-Fi, визначення вторгнень, мережна ідентифікація, безпека на основі місцезнаходження, відслідковування активів та гостьовий доступ

Управління мережею
Такий же рівень безпеки, масштабує мості, простоти розгортання й управління як й для дротових мереж

Уніфікована мережа
Архітектура на основі контролерів. Інтеграція з комутаторами та маршрутизаторами

Мобільна платформа
Доступ до корпоративної мережі як всередині приміщень, так й ззовні. Збільшення продуктивності труда співробітників. Перевірена платформа.

Клієнтські пристрої
90% Wi-Fi чипів сертифіковані як сумісні з обладнанням Ciscoю Вбудовані засоби забезпечення безпеки

Рисунок 3.1 – Структурна схема системи

Розгортання рішення Cisco Unified Wireless Network

П'ять елементів Cisco Unified Wireless Network мають фундаментальне значення для побудови захищених, успішних мереж WLAN корпоративного класу. Замовники мають можливість вибрати ті елементи й продукти Cisco Unified Wireless Network, які щонайкраще відповідають їхнім потребам у плані бездротових мереж. Замовники можуть почати з вибору клієнтських пристроїв і мобільної платформи, що включає в себе автономні й "полегшені" точки доступу, а потім, у міру появи нових вимог до бездротової мережі, доповнювати наявні елементи новими.

Рішення Cisco є сервісно-орієнтованим. Крім повної підтримки стандартів бездротового зв'язку 802.11a / b / g дане рішення передбачає підтримку додаткових сервісів, таких як передача голосу, системи запобігання вторгнень, контроль доступу до мережі (Network Admission Control, NAC), Cisco Self-Defending Network, динамічне керування радіочастотами, сервіси виявлення з високим дозволом і гостьовий доступ. Ці додаткові сервіси можуть бути впроваджені негайно або протягом певного періоду часу в процесі поетапної інтеграції. Замовники самостійно ухвалюють рішення щодо тім, коли і яким образом впроваджувати ці додаткові сервіси, виходячи із проектних графіків і організаційних вимог.

Більше докладна інформація про кожне з елементів рішення Cisco Unified Wireless Network приводиться нижче.

Клієнтські пристрої

Більше 95% сучасних ноутбуків підтримують технологію Wi-Fi. Крім того, існує широкий спектр спеціалізованих клієнтських пристроїв з підтримкою Wi-Fi, які призначені для застосування в різних галузях. Щоб задовольнити потреби корпорацій у мережах WLAN, клієнтські пристрої повинні взаємодіяти в захищеному режимі із провідними інфраструктурами WLAN і постійно забезпечувати ту функціональність, що необхідна організаціям.

Для впровадження рішення Cisco Unified Wireless Network ми настійно рекомендуємо використовувати або клієнтські пристрої, сумісні з устаткуванням Cisco і клієнтськими пристроями Cisco Aironet, або безпосередньо клієнтські пристрої Cisco Aironet. Більше 90% клієнтських пристроїв, що поставляються нині, мають сертифікат сумісності з устаткуванням Cisco, тому практично будь-який обраний клієнтський пристрій буде підтримувати потужні додаткові функції Cisco.

Завдяки перевірці сумісності із продукцією сторонніх виробників програма Cisco Compatible Extensions допомагає забезпечити наявність величезного вибору клієнтських пристроїв, пропонованих різними

постачальниками, які можуть взаємодіяти з інфраструктурою WLAN, побудованої на платформі Cisco. На цей момент більше 300 бездротових пристроїв мають сертифікат сумісності з Cisco (Cisco Compatible), і їхня кількість постійно росте. Більше 90% сучасних ноутбуків сертифіковані на предмет сумісності з Cisco.

Демонструючи свою прихильність передовим інноваціям і прагнучи задовольнити різноманітні вимоги замовників до корпоративних додатками, Cisco реалізує сучасні функції до прийняття офіційного стандарту через програму Cisco Compatible Extensions. Cisco надає своїм партнерам можливість відновлення програмного забезпечення для мобільних пристроїв, сумісних з Cisco, щоб забезпечити захист інвестицій і допомогти компаніям знайти варіант переходу до майбутніх галузевих стандартів і майбутньої функціональності інфраструктури Cisco WLAN. Керівники служб IT можуть із упевненістю розгортати мережі WLAN, навіть якщо вони обслуговують клієнтські пристрої різних типів, за умови їхньої сумісності з устаткуванням Cisco.

Жоден інший постачальник устаткування для мереж WLAN не зможе запропонувати можливості скористатися перевагами завтрашнього дня вже сьогодні. Тільки за допомогою Cisco можна вже зараз із упевненістю створювати потужні, масштабовані, захищені й керовані рішення. Завдяки програмі Cisco Compatible Extensions компанія Cisco може запропонувати функціональність WLAN наступного покоління вже сьогодні.

Рішення Cisco Unified Wireless Network підтримує також і клієнтів, сертифікованих на предмет відповідності стандартам Wi-Fi і IEEE 802.11, але ми все-таки рекомендуємо використовувати як клієнтські пристрої точки доступу Cisco Aironet або пристрою, сумісні з устаткуванням Cisco (Cisco Compatible), тому що вони підтримують інноваційні, перевірені Cisco передові функції.

Мобільна платформа

Організаціям потрібно, з однієї сторони реалізувати захищене підключення клієнтів WLAN по стандартах 802.11a / b / g за допомогою

відповідних точок доступу, а з іншого боку – забезпечити розгортання спеціалізованих ефірних / радіочастотних функцій, керування ними й контроль за їхньою роботою. Крім того, організаціям необхідні надійні рішення WLAN для організації підключення користувачів, що перебувають поза межами будинків і в кампусах, а також для організації зв'язку між будинками:

– Точки доступу й мости Cisco Aironet. Cisco пропонує цілий ряд автономних і "полегшених" точок доступу й бездротових мостів корпоративного класу, що дозволяють задовольнити найрізноманітніші вимоги до установки й умов роботи. Сімейство продуктів Cisco Aironet містить у собі найбільш широкий і гнучкий асортименти бездротових пристроїв, призначених для роботи у всіляких умовах – від комфортних офісів, устелених килимами, до самих жорстких умов або поза будинками.

– Точки доступу Cisco Aironet. Точки доступу Cisco Aironet забезпечують повсюдний доступ до мережі для найрізноманітніших бездротових середовищ, що діють як усередині будинків, так і за їхніми межами. Замовники можуть вибрати як "полегшені" точки доступу Cisco Aironet, так і точки доступу Cisco Aironet, що функціонують в автономному режимі.

– Cisco Aironet – це що прекрасно зарекомендувала себе, передова платформа, який належить більше 61% світового ринку. Вона є стандартом для корпоративних мереж WLAN. Точки доступу Cisco Aironet дозволяють організувати захищені, керовані й надійні бездротові підключення, що характеризуються винятково високою ємністю, дальністю дії й продуктивністю. Вони підтримують масу різноманітних варіантів установки, наприклад, з використанням одинарних і подвійних радіомодулів, убудованих або зовнішніх антен і захищених металевих корпусів. Точки доступу Cisco Aironet дозволяють досягти того рівня універсальності, ємності й безпеці корпоративного класу, що необхідний замовникам, що наміряються розгорнути в себе мережа WLAN. Ці точки доступу в стандартному варіанті поставки підтримують бездротові функції

plug-and-play, які дозволяють обійтися практично без втручання оператора в процесі налаштування. Наприклад:

– Точки доступу Cisco Aironet серій 1000 і 1130 AG ідеально підходять для офісів і аналогічних середовищ, що характеризуються приблизно однаковими умовами в межах всієї інфраструктури. Ці точки доступу постачені убудованими антенами, які мають передбачувану кругову діаграму спрямованості.

– Точки доступу Cisco Aironet серій 1240 AG ідеально підходять для більше складних радіочастотних середовищ, наприклад, для виробничих приміщень і складів, а також для просторів над фальш-стелями, які звичайно вимагають наявності зовнішніх антен і захищених металевих корпусів.

– "Полегшені" точки доступу Cisco Aironet серії 1500 для повнозв'язних (mesh) мереж пропонують економічний і масштабований варіант побудови захищених локальних бездротових мереж поза будинками, що забезпечують підключення користувачів, що перебувають у кампусах, приватний або публічний доступ для мобільних користувачів, що перебувають поза будинками.

Більшість точок доступу Cisco Aironet функціонують або на основі протоколу Lightweight Access Point Protocol (LWAPP), або в автономному режимі (без використання контролерів бездротової локальної мережі).

– Налаштування "полегшених" точок доступу Cisco Aironet і керування ними здійснюються в динамічному режимі, за допомогою протоколу LWAPP. Всі "полегшені" точки доступу Cisco Aironet підключаються до контролерів бездротової локальної мережі Cisco, тому замовники можуть використовувати різні сполучення точок доступу в рамках власної мережної інфраструктури. "Полегшені" точки доступу забезпечують радіочастотний доступ до бездротової мережі за допомогою унікальної розділеної архітектури керування доступом до середовища передачі (MAC), що передбачає розподіл сфер відповідальності: керування деякими критичними вчасно функціями здійснює безпосередньо точка доступу, а керування іншими функціями – контролер. Всі "полегшені" точки доступу Cisco Aironet підтримують додаткові сервіси, такі як швидкий,

захищений роумінг для голосових сервісів і сервісів виявлення, що дозволяє забезпечити видимість мережі в реальному часі. Сервіси виявлення й адміністрування підтримуються опціональними системами Cisco Wireless Location Appliance і Cisco Wireless Control System (WCS).

– Керування автономними точками Cisco Aironet здійснюється в індивідуальному порядку за допомогою програмного забезпечення Cisco IOS Software через інтерфейс командного рядка (CLI) або Web-Інтерфейс. Кожна автономна точка доступу є незалежною й не вимагає для нормальної роботи наявності контролера бездротової мережі або якого-небудь додаткового встаткування. Керування роботою автономних точок доступу здійснюється за допомогою CiscoWorks Wireless LAN Solution Engine (WLSE) або CiscoWorks WLSE Express. CiscoWorks WLSE являє собою системний додаток для керування роботою точок доступу й мостів Cisco Aironet, що функціонують в автономному режимі. Точки доступу Cisco Aironet, що функціонують в автономному режимі, можуть виступати в якості одного з компонентів рішення Cisco Unified Wireless Network. Однак для того, щоб скористатися всіма додатковими функціями й перевагами рішення Cisco Unified Wireless Network, замовники повинні модернізувати діючі автономні точки доступу Cisco Aironet, реалізувавши підтримку протоколу LWAPP, і перейти на роботу з використанням контролерів бездротової мережі.

– Бездротові мости Cisco Aironet. Безпроводні мости Cisco Aironet установлюють нові стандарти бездротових мостових з'єднань, забезпечуючи високопродуктивне й багатофункціональне рішення для об'єднання різних локальних мереж у рамках єдиної мережі рівня MAN або інфраструктури публічного доступу. Ці інноваційні мости пропонують гнучке, зручне в експлуатації рішення, що відповідає вимогам безпеки, пропонованим фахівцями в області мережних технологій WAN. Вони підтримують як з'єднання типу "точка-точка", так і з'єднання типу "точка-безліч точок", володіють однієї із кращих у

галузі дальністю дії й підтримують швидкість передачі даних до 54 Мбіт/с. Наприклад:

– Зовнішні точки доступу / мости Cisco Aironet серії 1300 можна використовувати як автономні точки доступу, мостів або мостів для робочих груп. Ці пристрої постачені корпусами підвищеної захищеності й забезпечують високошвидкісні, економічні бездротові з'єднання між стаціонарними або мобільними клієнтами й мережами.

– Бездротові мости Cisco Aironet серії 1400 дозволяють реалізувати високошвидкісні, високопродуктивні мостові з'єднання між точками, розташованими поза будинками в межах прямої видимості. Вони постачені корпусами підвищеної захищеності, які оптимізовані для роботи в досить суворих умовах на відкритому повітрі, що характеризуються більшими перепадами температур.

Уніфікація мережі

Процес інтеграції провідних і бездротових мереж має вирішальне значення для керування уніфікованими мережами, забезпечення їхньої масштабованості, захищеності й надійності. Для підтримки бездротових додатків корпоративного класу бездротові мережі повинні підтримувати такі загальносистемні функції, як політики безпеки, засобу запобігання вторгнень, керування радіочастотами, керування якістю обслуговування (QoS) і мобільність. Крім того, бездротові мережі повинні передбачати механізм плавної інтеграції з діючими корпоративними мережами.

Контролери Cisco для бездротової локальної мережі. Рішення Cisco підтримує мережну інфраструктуру, що без проблем функціонує на різних платформах. Для бездротових локальних мереж це рішення пропонує той же рівень захищеності, масштабованості, надійності, зручності в експлуатації й керованості, що й для провідних локальних мереж. Воно є надійним варіантом переходу на всі основні комутуючі й маршрутизуючі платформи Cisco за допомогою контролерів Cisco для бездротових локальних мереж. Cisco є єдиним

у світі виробником, що пропонує повноцінне, всеосяжне рішення, що характеризується високим рівнем уніфікації й інновацій, і забезпечує надійний захист інвестицій, дозволяючи створити безпечний, мобільний, інтерактивний робочий простір для бездротових і провідних мереж.

Сьогодні уніфікація провідних і бездротових мереж здійснюється за допомогою модулів Cisco Catalyst 6500 Series Wireless Services Module (WiSM) і Cisco Wireless LAN Controller Module для маршрутизаторів з інтеграцією сервісів. Модуль Cisco WiSM забезпечує неперевершений рівень захищеності, мобільності, резервування й зручності в експлуатації критично важливих для бізнесу бездротових мереж. Він розрахований на середні й великі корпоративні будинки, що дозволяють поєднувати в кластери до 3600 "полегшених" точок доступу на один роумінговий домен. Один модуль підтримує до 300 "полегшених" точок доступу й більше 10.000 бездротових клієнтських пристроїв. Модуль Cisco Wireless LAN Controller для маршрутизаторів з інтеграцією сервісів дозволяє підприємствам малого й середнього бізнесу будувати економічні й захищені бездротові мережі у філіях. Він одночасно підтримує до 6 "полегшених" точок доступу Cisco і може використовуватися в маршрутизаторах з інтеграцією сервісів Cisco серій 2800 / 3800 і 3700.

Крім того, Cisco пропонує два автономних контролери бездротової локальної мережі – пристрою Cisco серій 4400 і 2000. Ці автономні контролери повністю інтегруються в мережну інфраструктуру й служать для реалізації додаткових функцій безпеки, виконання політик керування якістю обслуговування (QoS) і підтримки інших додаткових сервісів. Ємність цих контролерів варіюється від 6 точок доступу для серії 2000 до 100 точок доступу для серії 4400. Ці контролери можна об'єднати в кластер, що дозволить забезпечити підтримку до 2400 точок доступу на один роумінговий домен.

Контролери Cisco для бездротової мережі відповідають за реалізацію загальносистемних функцій бездротової мережі, таких як убудована система захисту від вторгнень (IPS), засобу керування радіочастотами в реальному часі,

механізм розгортання без участі оператора й резервування за схемою N+1. Взаємодія контролерів з "полегшеними" точками доступу й пристроєм керування дозволяє досягти високого рівня продуктивності й забезпечити підтримку додаткових можливостей керування. Контролери Cisco для бездротових мереж дозволяють забезпечити той рівень керованості, масштабованості, безпеці й надійності, що необхідний мережним адміністраторам для побудови захищених бездротових мереж корпоративного класу, призначених для організацій різного масштабу, починаючи від філій і невеликих підприємств і закінчуючи великими кампусами.

Мережне адміністрування

Мережним адміністраторам необхідний надійний і економічний інструмент для планування, налаштування параметрів і адміністрування бездротових локальних мереж. Цей інструмент повинен працювати в централізованому режимі, забезпечувати спрощене керування й володіти зручним у використанні графічним інтерфейсом.

Cisco Wireless Control System. Cisco пропонує інтерфейс світового класу для адміністрування мереж WLAN – передову систему Cisco Wireless Control System (WCS). Завдяки підтримці централізованого керування Cisco WCS значно полегшує процес адміністрування бездротової мережі. Ця платформа створює надійну основу, що дозволяє мережним адміністраторам проектувати корпоративні бездротові мережі, вести їхній моніторинг і управляти ними в централізованому режимі. Таким чином, пропонована платформа дозволяє спростити процес експлуатації мережі й знизити сукупну вартість володіння нею.

Рішення Cisco WCS уже сьогодні пропонує надійну й зручну в роботі платформу для адміністрування критично важливих для бізнесу бездротових мереж. Дана платформа дозволяє планувати й проектувати бездротові локальні мережі, здійснювати керування радіочастотами, стежити за переміщеннями абонента, контролювати роботу системи захисту від вторгнень, набудовувати параметри систем WLAN, здійснювати їхній моніторинг і адміністрування. Вона

дозволяє з легкістю управляти безліччю контролерів і пов'язаних з ними "полегшених" точок доступу. Вона підтримує механізм розгортання, не потребуючої участі оператора, і має зручні графічні інтерфейси, які дозволяють полегшити й здешевить процес впровадження й експлуатації бездротової мережі. У процесі повсякденного керування мережами досить корисні будуть детальні аналітичні звіти й засоби аналізу тенденцій. В особі Cisco WCS мережні адміністратори одержують універсальний інструмент, що дозволяє вирішувати такі завдання, як радіочастотне планування, застосування політик, оптимізація мережі, усунення проблем, спостереження за переміщеннями користувачів, моніторинг безпеки й адміністрування бездротових локальних мереж.

Уніфіковані додаткові сервіси

Передова мережа WLAN повинна підтримувати нові мобільні додатки, нові технології Wi-Fi і додаткові засоби виявлення й запобігання погроз. Крім того, сервіси такої підтримки повинні бути економічними, простими в розгортанні й експлуатації.

Cisco Unified Wireless Network Advanced Services. Рішення Cisco забезпечує уніфіковану підтримку передових додатків. Ця підтримка була споконвічно убудована в рішення Cisco і не є результатом наступної доробки. Рішення Cisco є сервісно-орієнтованим і містить у собі цілий ряд різних сервісів, які можна розгортати або відразу, або поступово, у процесі поетапного впровадження. Організації можуть вибірково впроваджувати необхідні сервіси й додатки, виходячи із власних індивідуальних потреб. Додаткові сервіси Cisco є передовими, інноваційними й всеосяжними.

Рішення Cisco Unified Wireless Network підтримує нові мобільні додатки, нові технології Wi-Fi і додаткові засоби виявлення й запобігання погроз. Воно підтримує такі потужні функції, як бездротові сервіси передачі голосу VoIP і спостереження за переміщенням абонента, а також додаткові функції забезпечення безпеки бездротової мережі, такі як NAC, Cisco Self-Defending Network і гостьовий доступ.

– VoIP у бездротовій мережі. Підтримка VoIP у бездротовій мережі дозволяє організаціям реалізувати економічні сервіси голосового зв'язку в режимі реального часу, використовуючи наявну бездротову інфраструктуру. Компанії зможуть скористатися функціями VoIP завдяки потужним можливостям голосового зв'язку, якими володіє телефон Cisco Wireless IP Phone 7920. Цей телефон з убудованою підтримкою Wi-Fi підтримує також і ряд інтелектуальних сервісів, таких як безпека, мобільність, керування якістю обслуговування (QoS) і мережне адміністрування в масштабі всієї мережі Cisco.

– Сервіси виявлення. Сервіси виявлення високого дозволу підтримують критично важливі додатки, такі як моніторинг переміщень коштовних ресурсів, керування інфраструктурою IT і забезпечення безпеки залежно від географічного положення. Апаратне рішення Cisco Wireless Location Appliance серії 2700 дозволяє одночасно відслідковувати місце розташування тисяч авторизованих і неавторизованих активних пристроїв Wi-Fi з точністю до декількох метрів безпосередньо за допомогою інфраструктури WLAN. Це рішення дозволяє реалізувати економічні сервіси виявлення з високим дозволом для критично важливих додатків, таких як моніторинг переміщень коштовних ресурсів, керування інфраструктурою IT і забезпечення безпеки залежно від географічного положення. За допомогою цього інноваційного пристрою можна інтегрувати широкий спектр технологічних рішень і прикладних розробок компаній-партнерів за допомогою багатофункціонального відкритого інтерфейсу прикладного програмування (API) для розгортання нових важливих бізнес-додатків. Варто також помітити, що, якщо сервіси виявлення Cisco використовуються в сполученні з бездротовими сервісами VoIP, користувачі можуть скористатися зв'язком e911 для надзвичайних ситуацій.

– NAC. NAC – це набір технологій і рішень, в основі яких лежить галузева ініціатива, запропонована компанією Cisco Systems. Мережі Cisco WLAN підтримують NAC, використовуючи мережну інфраструктуру для забезпечення політики безпеки для всіх бездротових пристроїв, що намагаються

одержати доступ до обчислювальних мережних ресурсів. Це дозволяє обмежити збиток від різноманітних погроз безпеки, таких як віруси, хробаки й шпигунське програмне забезпечення. Мережі Cisco WLAN підтримують як NAC Appliance (Cisco Clean Access), так і NAC Framework.

– Cisco Self-Defending Network. Стратегія мережі, що самообороняється, Cisco Self-Defending Network відбиває подання Cisco про те, як повинна бути реалізована система безпеки інтегрованої мережі. Ця стратегія допомагає організаціям ідентифікувати й запобігати як відомі, так і невідомі погрози безпеки, а також адаптуватися до них. Мережі Cisco WLAN підтримують інтеграцію з рішенням Cisco Self-Defending Network, що дозволяє підтримувати безпека в масштабі всієї мережі й забезпечити наданя доступу до мережних ресурсів на основі ідентифікації користувачів.

– Гостьовий доступ. Гостьовий доступ дозволяє замовникам зберегти безпеку своєї бездротової мережі в недоторканності й одночасно надати своїм клієнтам, постачальникам і партнерам можливість контрольованого доступу до мережі WLAN замовника.

Cisco продовжить підтримувати й розвивати додаткові сервіси й функції, покликані допомогти замовникам у пошуку мобільних рішень, що дозволяють справлятися із завданнями бізнесу сьогодні й у майбутньому.

Функціональні особливості

Потужні функціональні можливості продуктів, що є складеними компонентами рішення Cisco Unified Wireless Network, спричиняються підтримку цілого ряду передових функцій і переваг. Це потужне рішення забезпечує можливості централізованого керування й контролю за всією інфраструктурою.

– Безпека. Фундаментальна оптимальна методика забезпечення безпеки бездротової локальної мережі припускає можливість захисту радіочастотного середовища й контролю за її використанням. Компанія Cisco займає провідні позиції в сфері забезпечення радіочастотного захисту корпоративного класу й

моніторингу політик безпеки мережі WLAN. До числа функцій безпеки бездротових мереж Cisco ставляться:

– Контрольований доступ до мережі WLAN з використанням цілого ряду політик автентифікації й шифрування, у тому числі стандартів 802.11i, Wi-Fi Protected Access (WPA) і WPA2, а також мобільних віртуальних часток мереж VPN

– Система запобігання вторгнень WLAN IPS, що дозволяє виявляти й блокувати шахрайські точки доступу, неасоційовані клієнтські пристрої й однорангові (ad-hoc) мережі, а також використовувати набудовуються сигнатури, що, радіочастотних атак для захисту від погроз загального характеру, яким піддаються бездротові мережі

– Наявність захищених засобів керування границями безпеки інфраструктури й радіочастотного рівня

– Адміністрування. Cisco спрощує процес адміністрування мереж WLAN завдяки наявності засобів, що забезпечують повну видимість і керованість радіочастотного середовища. Це дозволяє підвищити масштабованість мережі, прискорює процес усунення проблем і підвищує продуктивність роботи мережних адміністраторів. В остаточному підсумку все це приводить до зниження операційних витрат. До числа функцій адміністрування Cisco ставляться:

– Спрощений механізм адміністрування й експлуатації WLAN, що дозволяє демистифікувати радіочастотні питання й значно полегшити керування радіочастотним середовищем.

– Функції радіочастотного сканування, керування й моніторингу, убудовані безпосередньо в інфраструктуру WLAN, дозволяють одержати самоналаштувальну, самооптимізуючуюся й бездротову мережу, що самовиліковує.

– Одночасний моніторинг місцезнаходження тисяч пристроїв безпосередньо з інфраструктури WLAN з використанням технології

радіочастотної "дактилоскопії" (fingerprinting) від Cisco, на яку оформлена патентна заявка.

– Потужні інструмент планування, розгортання й адміністрування WLAN.

– Потужні засоби усунення проблем і діагностики, що дозволяють звістки моніторинг продуктивності й збоїв у режимі, що попереджає, у тому числі графічні теплові карти для спрощеного аналізу.

– Централізовані процесори політик, які полегшують налаштування й застосування політик безпеки й керування якістю обслуговування (QoS) на загальносистемному рівні.

– Якість роботи. Зв'язок у зоні покриття WLAN повинна бути надійної, а радіочастотний діапазон повинен бути оптимізований для забезпечення максимальної якості роботи WLAN. Cisco досягає цього завдяки наступним функціональним особливостям:

– Підтримка механізмів керування якістю обслуговування (QoS) для голосових і чутливих до затримок додатків, у тому числі підтримка угод про виділення пропускної здатності через бездротову мережу.

– Керування ємністю системи в реальному часі з використанням балансування навантаження.

– Висока ємність і універсальність: система може працювати як у комфортних офісах, так і в умовах, що вимагають підвищеної захищеності, у широкому температурному діапазоні.

– Мережі, що самовиліковують, WLAN, що забезпечують високий рівень доступності, зокрема, за рахунок виявлення й усунення "дір" у зоні покриття.

– Мобільність. Кінцевим користувачам необхідний безперебійний доступ до мережі в процесі роумінгу між точками доступу (усередині підмереж і між ними). Рішення Cisco WLAN реалізує наступну функціональність:

– Захищений роумінг на рівнях 2 і 3.

– Мережі VPN типу "випливай за мною", які дозволяють клієнтам користуватися тунелями VPN у роумінзі.

– Швидкий, захищений і масштабований роумінг у середовищах 802.11i.

– Передача контексту політик безпеки й керування якістю обслуговування (QoS), що дозволяє проводити ідентифікацію користувачів у роумінзі.

– Бездротовий зв'язок без границь, усередині будинків і поза ними, у тому числі динамічні бездротові повнозв’язні (mesh) мережі.

– Масштабованість. Бездротова мережа повинна бути досить масштабованої для підтримки поточних і майбутніх вимог бізнесу. Cisco пропонує:

– підтримку мереж WLAN у кампусах, філіях, на віддалених вузлах і на відкритому просторі (поза будинками);

– підтримку десятків, сотень і тисяч централізованих і віддалених точок доступу;

– надійність, резервування й стійкість до збоїв мереж WLAN.

– Інтеграція. Наскрізна інтеграція провідних і бездротових мереж знижує сукупну вартість володіння. Рішення Cisco мінімізує сукупну вартість володіння мережами WLAN завдяки наступним факторам:

– Уніфікована інфраструктура для провідних і бездротових мереж, що дозволяє централізовано управляти всім трафіком WLAN.

– Реалізація потужної, інтелектуальної функціональності, властивої провідним інфраструктурам на платформі Cisco, для бездротового трафіку, зокрема, таких функцій, як керування якістю обслуговування (QoS) і політики адміністрування.

– Підтримка декількох типів серверів автентифікації, авторизації й обліку (authentication, authorization, and accounting, AAA).

– Інтеграція клієнта із програмою Cisco Compatible Extensions.

– Легкий перехід від автономних точок доступу Cisco Aironet до точок, що працюють на основі протоколу LWAPP.

– Сервіси / додатка. Корпоративна мережа WLAN повинна підтримувати широкий спектр потужних сервісів і додатків, які використовують унікальні можливості мобільного зв'язку. Рішення Cisco підтримує наступні можливості:

– Голосові сервіси, реалізовані через програмний додаток або слухавку.

– Сервіси спостереження з високим дозволом за місцем розташування користувачів і ресурсів.

– Гостьовий доступ для клієнтів, консультантів, підрядників, постачальників і виробників.

– Пристрої спеціалізованого застосування (application-specific devices, ASD), наприклад, пристрою, використовувані в роздрібній торгівлі або на виробництві.

Переваги

Рішення Cisco Unified Wireless Network дозволяє компаніям із упевненістю впроваджувати бездротові мережі. Рішення Cisco Unified Wireless Network обіцяє бізнесу цілком реальні й відчутні переваги. Ці переваги сполучаються із застосуванням методик захисту мережі, що дозволяють забезпечити безпека корпоративного класу. До числа переваг Cisco Unified Wireless Network ставляться:

– Зниження сукупної вартості володіння (TCO). Рішення Cisco Unified Wireless Network знижує сукупну вартість володіння завдяки мінімізації операційних і капітальних витрат, пов'язаних з розгортанням бездротової мережі й керуванням нею. Рішення Cisco пропонує:

– масштабовану, безпроблемну схему адміністрування, що не приводить до збільшення навантаження на персонал відділу IT;

– контроль за витратами на впровадження бездротової мережі, що не змушує жертвувати надійністю;

– бюджетні бездротові мости типу "точка-точка" і "точка-безліч точок", спеціально спроектовані для забезпечення зручності установки й експлуатації;

– Поліпшені видимість і контроль WLAN. Cisco забезпечує більшу прозорість і більше чіткий контроль за роботою бездротової мережі LAN, які допомагають забезпечити безпечну й надійну роботу бездротових додатків у масштабі всього підприємства й реалізувати централізований механізм керування, що дозволяє підвищити рівень масштабованості й полегшити процес експлуатації.

– Дане рішення дозволяє одночасно відслідковувати місце розташування тисяч авторизованих і неавторизованих активних пристроїв Wi-Fi з точністю до декількох метрів безпосередньо за допомогою інфраструктури WLAN.

– Архітектура системи має убудований механізм забезпечення стійкості й підтримує централізовану схему контролю й керування.

– Бездротові пристрої plug-and-play, не потребуючі втручання оператора в процесі налаштування.

– Динамічне керування радіочастотами. Cisco є лідером в області інноваційних радіочастотних рішень; компанія створює інтелектуальні рішення WLAN, цінність яких базується на унікальних властивостях радіотехнології.

– Виявлення змін у радіочастотному середовищі й динамічній адаптації до цих змін у реальному часі.

– Інтелектуальна панель керування радіочастотами, що дозволяє виконувати автоматичне налаштування, автоматичне усунення несправностей і автоматичну оптимізацію мережі.

– Потужний механізм забезпечення безпеки WLAN і захисту мережі. Інтегрована система захисту WLAN від вторгнень (IPS) запобігає появі проломів у системі безпеки й перешкоджає виникненню незахищених з'єднань WLAN, які ставлять під погрозу безпека всієї мережі.

– Можливість використання файлів сигнатур, що набудовуються, атак для швидкого виявлення й стримування найпоширеніших радіочастотних атак, таких, як Netstumbler, FakeAP і void11.

– Технологія радіочастотного "дактилоскопірування" (fingerprinting) підтримує високоточне виявлення місця розташування пристроїв.

– Технології Cisco Self-Defending Network і NAC обмежують можливий збиток від різноманітних погроз безпеки, таких як віруси, хробаки й шпигунське програмне забезпечення.

– Блокування шахрайських точок доступу й клієнтів, незалежно від того, підключені вони до провідного або до бездротової мережі, дозволяє підтримувати на належному рівні безпека мережі й запобігає доступу несанкціонованих користувачів до корпоративних ресурсів.

– Сумісні з устаткуванням Cisco клієнтські пристрої сприяють виявленню шахрайських підключень до ефірного / радіочастотній середовищу.

– Уніфікація провідних і бездротових мереж. Cisco є єдиним у світі виробником, що пропонує повноцінне, всеосяжне рішення, що характеризується високим рівнем уніфікації й інновацій, і забезпечує надійний захист інвестицій, дозволяючи створити безпечний, мобільне, інтерактивне робочий простір для бездротових і провідних мереж.

– Можливість створення й застосування політик автентифікації й керування доступом.

– Тепер ті політики, якими можна було користуватися для забезпечення безпеки корпоративних провідних мереж (NAC, міжмережні екрани), керування ними (кореляція подій, віртуальні локальні мережі, керування політиками) і забезпечення якості обслуговування QoS (802.1p, керування пропускною здатністю), стали доступні й у бездротових мережах.

– Надійна стратегія міграції, заснована на інтеграції з вибраними маршрутизаторами й комутаторами Cisco, забезпечує можливість побудови потужної, інтегрованої корпоративної мережі.

– Мобільність для підприємства. Мобільність стає ключовим елементом сучасного підприємства завдяки рішенням компанії Cisco, що допомагають адміністраторам IT з легкістю розвертати захищені, критично важливі для бізнесу

мережі WLAN, що доповнюють можливості наявної мережної інфраструктури – як усередині будинків, так і за їхніми межами.

– Універсальність, висока ємність, захищеність і функціональні можливості корпоративного класу – все те, що потрібно замовникам мереж WLAN.

– Передача контексту дозволяє ідентифікувати користувачів у процесі їхнього роумінгу між доменами рівнів 2 і 3.

– Підтримка механізмів керування якістю обслуговування (QoS) і мультимедійних можливостей Wi-Fi для технології VoIP.

– Зручні в установці бездротові мости.

– Збільшення продуктивності, створення більше сприятливих умов для співробітництва й підвищення швидкості реагування. Технологія Cisco допомагають співробітникам організації домагатися більшого від кожної ділової зустрічі, швидше приймати рішення й більш ефективно використовувати щохвилини незалежно від того, проводите Ви її в офісі або в шляху.

– Установи охорони здоров'я зможуть підвищити якість обслуговування пацієнтів.

– Університети й освітні установи зможуть відкрити нові варіанти співробітництва між студентами й викладачами.

– Фінансові установи зможуть одержати доступ до даних про клієнтів у реальному часі.

– Урядові організації зможуть забезпечити більше оперативний доступ до інформації й підвищити в такий спосіб рівень суспільної безпеки.

– Виробничі підприємства зможуть використовувати дані, що надходять у реальному часі із цехів, для підтримки адаптивних (just-in-time) процесів виробництва й складаня.

– Підприємства роздрібної торгівлі зможуть забезпечити повну мобільність даних у масштабі всього складу або магазина, що дозволить підвищити ефективність обслуговування клієнтів.

– Компанії зможуть організувати для своїх співробітників, що перебувають у дорозі, доступ до корпоративних мереж за допомогою мереж публічного доступу WLAN.

– Корпорації зможуть більш ефективно відслідковувати місцезнаходження ресурсів і аналізувати критично важливу для бізнесу інформацію. Крім того, вони зможуть підвищити продуктивність співробітників завдяки можливості обміну інформацією в реальному часі.

Уніфікація, інновації й захист інвестицій Cisco

Будучи світовим лідером в області технологій WLAN, компанія Cisco пропонує саму широку в галузі лінійку продуктів для побудови корпоративних мереж WLAN. Cisco продовжить роботу над розвитком технології WLAN з метою створення нового покоління корпоративних мереж. Cisco і надалі буде розвивати концепцію Cisco Unified Wireless Network, прагнучи, з одного боку, забезпечити підтримку нових додатків і просування технології WLAN, а з іншого боку – задовольнити потреби компаній у мережах корпоративного класу.

Технологічна уніфікація, інновації й захист інвестицій, пропоновані рішенням Cisco, допоможуть створити захищений, мобільне й інтерактивний робочий простір для організацій, що впроваджують мережі WLAN. Cisco допоможе забезпечити захист інвестицій замовника завдяки перевіреній схемі відновлення на місці убудованого програмного забезпечення, відновлення програмного забезпечення й ретельного аналізу нових вимог до встаткування. У майбутньому Cisco припускає ввести кілька варіантів впровадження, удосконалити програмне забезпечення, запропонувати нові можливості масштабування й функції захисту, удосконалити апаратну частину, а також забезпечити більше високий ступінь інтеграції провідних і бездротових мереж. Співробітничаючи з Cisco, замовники можуть бути впевнені в тому, що їхньої інвестиції в мережі WLAN будуть захищені не тільки сьогодні, але й завтра.

Програма фінансування Cisco WLAN

Щоб полегшити процес впровадження рішень Cisco WLAN, замовники можуть включити пакет фінансування бездротової мережі Cisco у програму лізингу Cisco Systems Capital® Corporation. Програма лізингу Cisco Systems Capital Corporation дозволяє компаніям мінімізувати обсяг початкових інвестицій, установити контроль над бюджетами й уникнути технологічного відставання. Одночасно із цім компанії зможуть підвищити продуктивність роботи співробітників, забезпечити більше оперативну їхню реакцію на запити клієнтів і створити більше сприятливі умови для спільної діяльності між співробітниками, партнерами й замовниками.

Короткий опис ситуації

Бурхливий ріст кількості ноутбуків з підтримкою Wi-Fi і клієнтських пристроїв, сумісних з устаткуванням Cisco, по усьому світі сприяє швидкому виникненню всі нових і нових мереж WLAN у корпоративних кампусах, у філіях і віддалених офісах. Організації, що впроваджують бездротові мережі WLAN, досягають підвищення продуктивності й ефективності роботи співробітників, здобувають додаткові конкурентні переваги й надають своїм користувачам новий рівень волі й гнучкості. Такі мережі WLAN дозволяють удосконалити бізнес-операції й забезпечити доступ у реальному часі до критично важливого для бізнесу додаткам і мережним ресурсам.

Рішення Cisco Unified Wireless Network дає можливість об'єднати провідні й бездротові мережі. Воно дозволяє ідентифікувати й запобігти як відомі, так і невідомі погрози безпеки, а також адаптуватися до них. Рішення Cisco Unified Wireless Network підтримує інноваційні радіочастотні технології й забезпечує для бездротових мереж той же рівень безпеки, масштабованості й керованості, до якого організації звикли в процесі експлуатації діючих провідних мереж.

Рішення Cisco має необхідний рівень гнучкості й масштабованості, що дозволяє задовольнити вимоги мереж будь-якого масштабу, починаючи від мереж невеликих підприємств і закінчуючи мережами найбільших мультинаціональних компаній; дане рішення дозволяє забезпечити повноцінний

бездротовий зв'язок на базі мереж WLAN у кампусах і філіях, в університетах, у роздрібній торгівлі, фінансових установах, на виробничих підприємствах і в установах охорони здоров'я, а також у місцях розміщення хот-спотів.

За допомогою рішення Cisco Unified Wireless Network компанії зможуть поставити радіочастотне середовище на службу своєму бізнесу. Це рішення пропонує відповіді на багато питань, пов'язані із впровадженням критично важливих для бізнесу мереж WLAN і керуванням ними, а також з налаштуванням радіочастотного середовища. Воно дозволяє знизити сукупні операційні витрати й спростити процеси розгортання й експлуатації мережі, а також керування нею. Це рішення допомагає мережним адміністраторам використовувати всі переваги вже наявних інструментів, знань і мережних ресурсів для максимально економічного рішення питань, пов'язаних із забезпеченням безпеки WLAN, її розгортання й керування.

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 наведена на рисунку 3.2.

У системі інтелектуальних сервісів протидії зловмисному впливу на мережу 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.

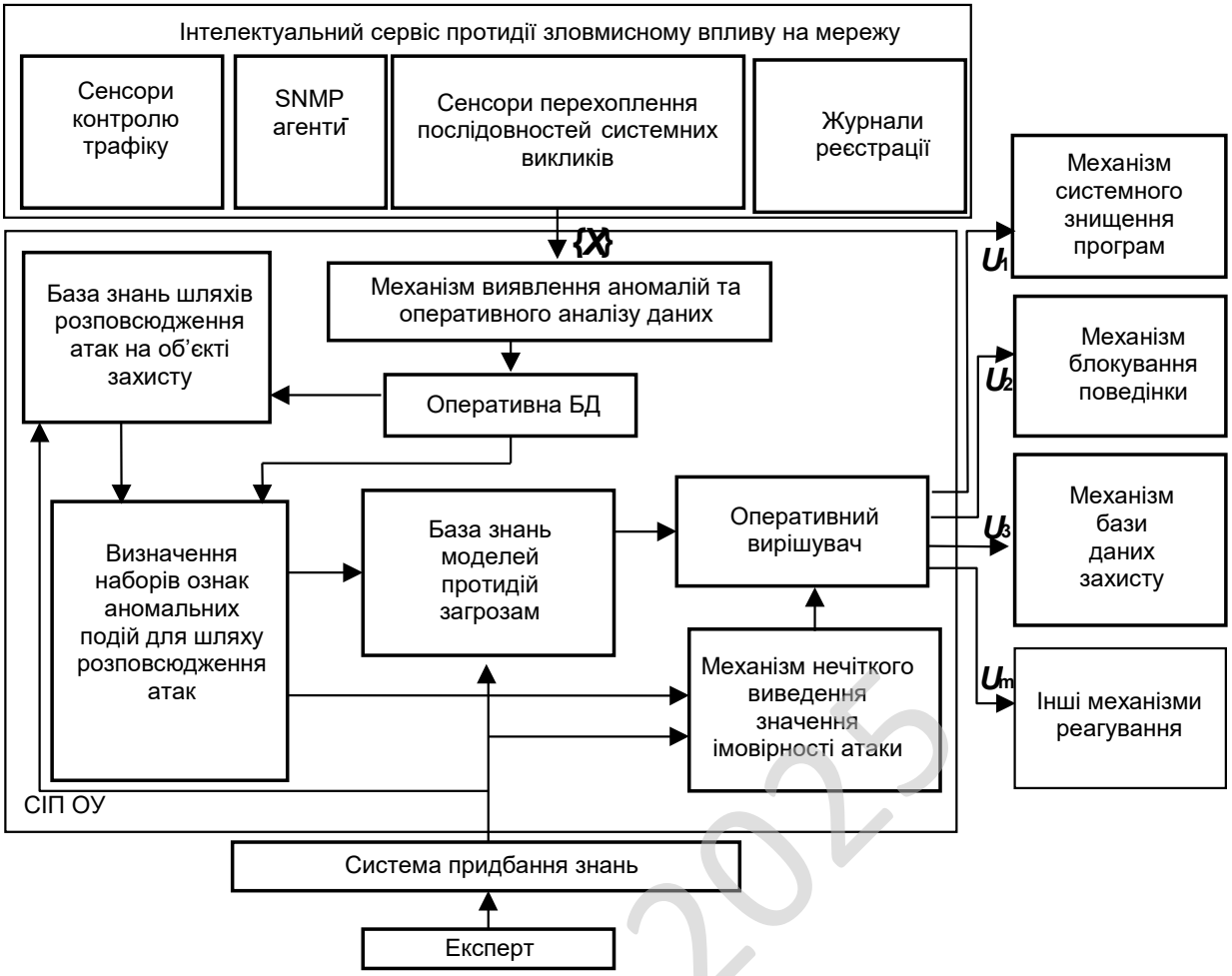


Рисунок 3.2 – Функціональна схема системи

Через необхідність максимальної структуризації розроблювальної системи й рішень, пропонується трирубіжна модель захисту, яка щонайкраще задовольняє всієї сукупності умов її розробки, експлуатації й удосконалення. Трирубіжна 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.

У процесі організаційно-технічного управління, планування ЗІ як функція управління 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»;

$W_{и_l}$ – 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 трирубіжної 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 завданя вибору задається морфологічною матрицею. Розробляються морфологічні матриці засобів захисту для три 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:

$$C_{\Sigma} \le C$$

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;

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

$C_{k_s}^{\text{В}}$ – 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. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

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

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

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

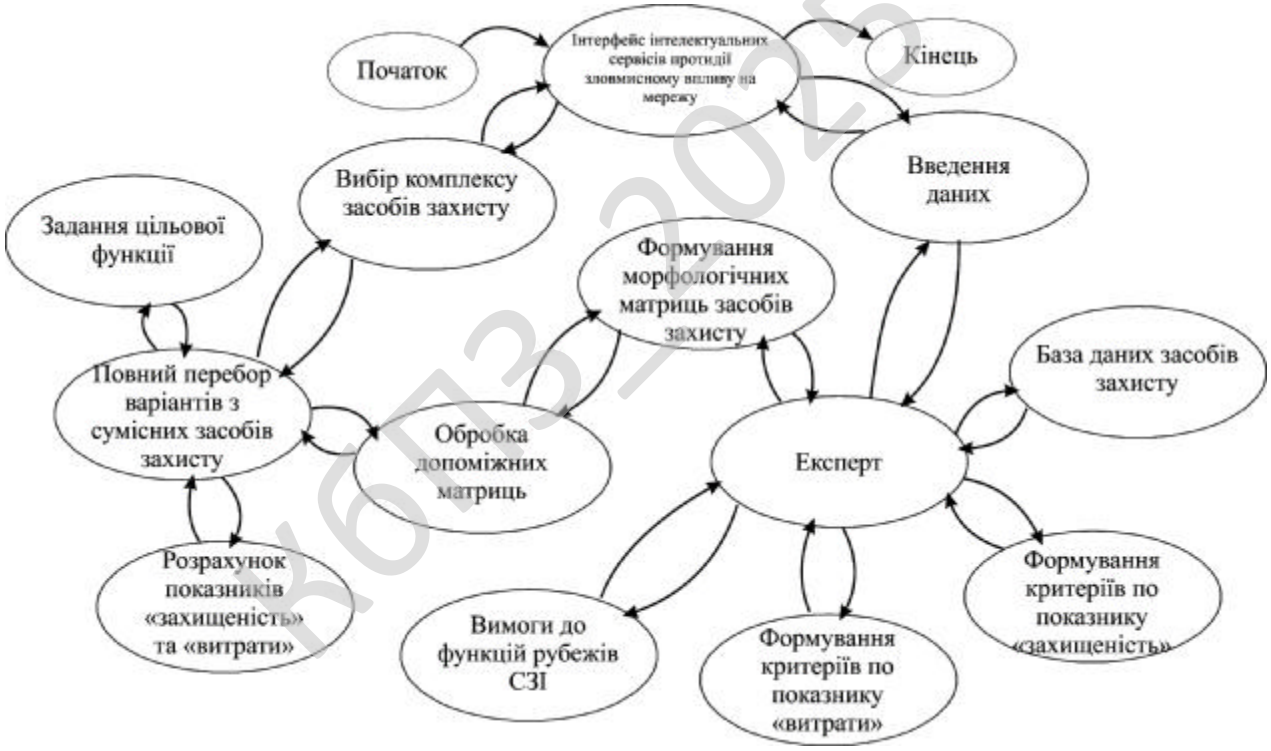


Рисунок 3.3 – Діаграма взаємодії процесів

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

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

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

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

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

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

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

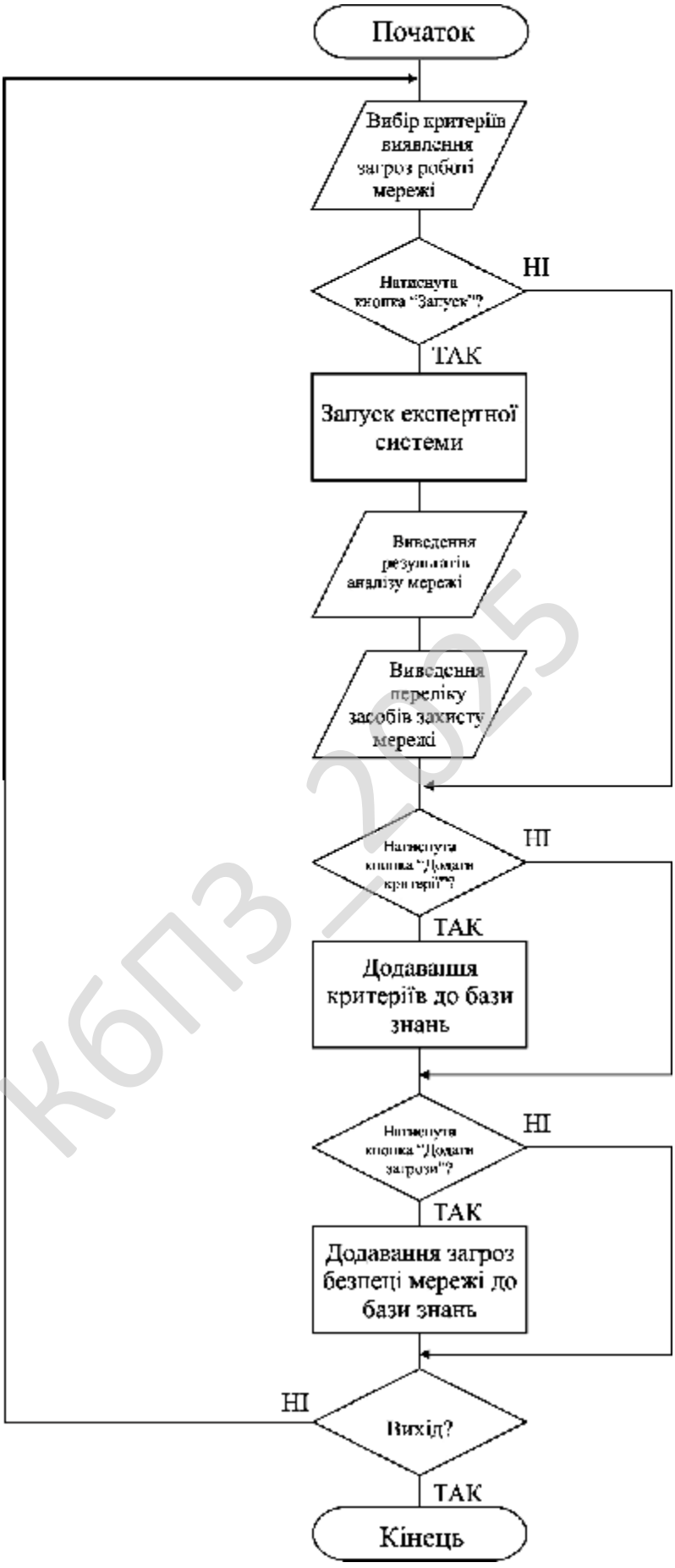


Рисунок 4.1 – Блок-схема основної програми

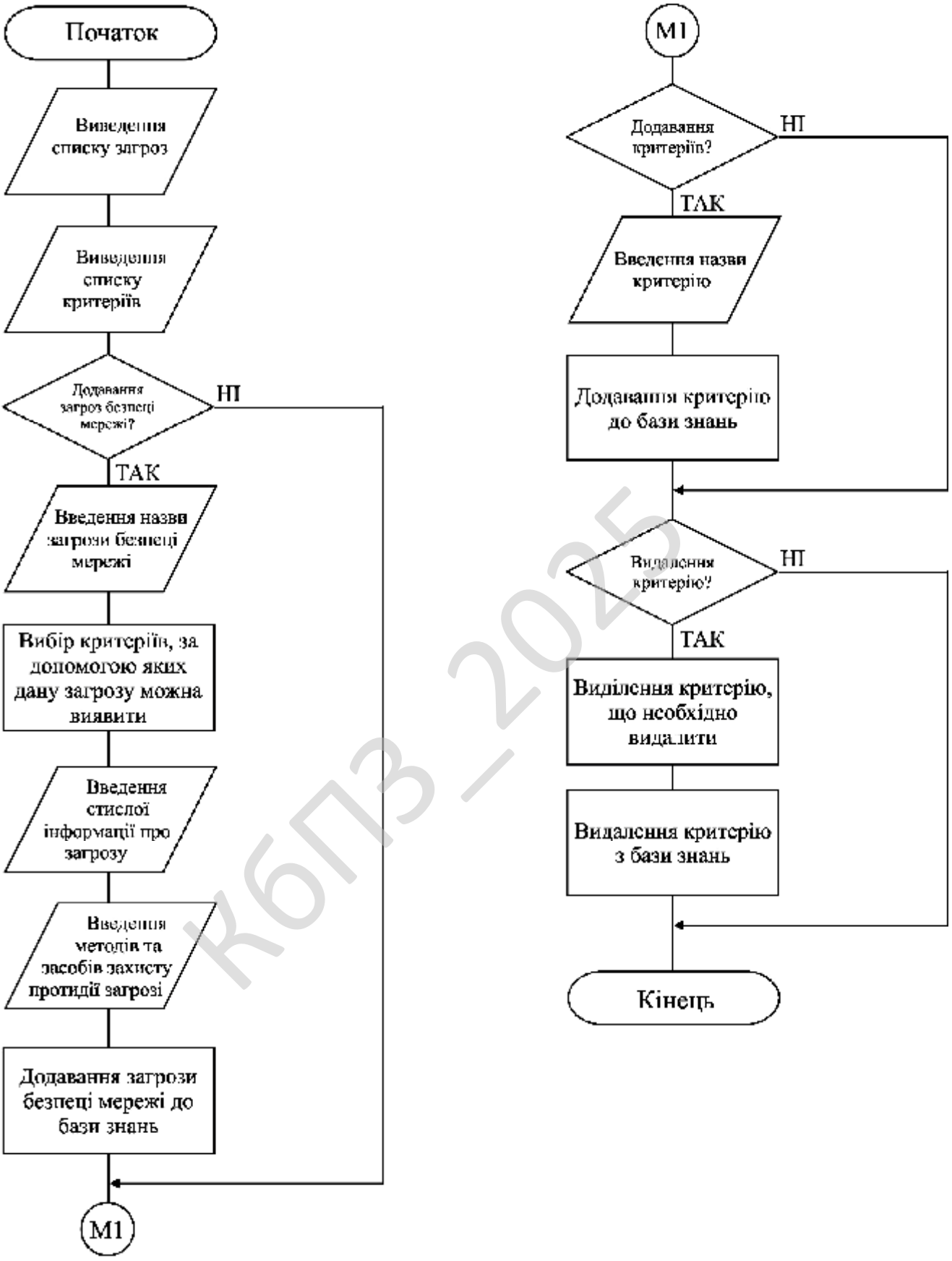


Рисунок 4.2 – Блок-схема роботи підпрограми

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

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

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу.

Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується.

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

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

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

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

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю інтелектуальних сервісів протидії зловмисному впливу на мережу.

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

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

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів.

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок.

Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

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

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

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

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

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

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

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

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

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

– Програмістам які реалізовують модулі інформаційної системи.

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

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

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

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

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

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме — за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати

діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

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

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

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

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

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

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

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

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

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

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

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

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та

об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

Peer-to-peer (рівний до рівного) – варіант архітектури системи, в основі якої стоїть мережа рівноправних вузлів.

Комп'ютерні мережі типу peer-to-peer (або P2P) засновані на принципі рівноправності учасників і характеризуються тим, що їх елементи можуть зв'язуватися між собою, на відміну від традиційної архітектури, коли лише окрема категорія учасників, яка називається серверами може надавати певні сервіси іншим.

Фраза «peer-to-peer» була вперше використана у 1984 році Парбауелом Йохнухуйтсманом (Parbawell Yohnuhuitsman) при розробці архітектури Advanced Peer to Peer Networking фірми IBM.

В чистій «peer-to-peer» мережі не існує поняття клієнтів або серверів, лише рівні вузли, які одночасно функціонують як клієнти та сервери по відношенню до інших вузлів мережі. Ця модель мережевої взаємодії відрізняється від клієнт-серверної архітектури, в якій зв'язок відбувається лише між клієнтами та центральним сервером.

Така організація дозволяє зберігати працездатність мережі при будь-якій конфігурації доступних її учасників. Проте практикується використання P2P мереж які все ж таки мають сервери, але їх роль полягає вже не у наданні сервісів, а у підтримці інформації з приводу сервісів, що надаються клієнтами мережі.

В P2P системі автономні вузли взаємодіють з іншими автономними вузлами. Вузли є автономними в тому сенсі, що не існує загальної влади, яка може контролювати їх.

В результаті автономії вузлів, вони не можуть довіряти один одному та покладатися на поведінку інших вузлів, тому проблеми масштабування та надмірності стають важливішими ніж у випадку традиційної архітектури.

Сучасні P2P-мережі набули розвитку завдяки ідеям, пов'язаними з обміном інформацією, які формувалися у руслі того, кожен вузол може надавати та отримувати ресурси які надаються будь-якими іншими учасниками. У випадку мережі Napster, це був обмін музикою, в інших випадках це може бути надання процесорного часу для пошуку інопланетних цивілізацій (SETI@home) або ліків від раку (Folding@home).

Переваги P2P

Розподіл/зменшення вартості. Централізовані системи, які обслуговують багато клієнтів, зазвичай складають більшість вартості системи. Коли, ця вартість стає дуже великою, архітектура P2P може допомогти розподілити вартість серед користувачів. Наприклад, серед систем файлообміну Napster дозволив розподілити вартість зберігання файлів і міг підтримувати індекс, потрібний для сумісного використання. Економія коштів, здійснюється за допомогою використання та об'єднання ресурсів, які в іншому випадку не використовуються (наприклад SETI@home). Оскільки вузли зазвичай є автономними, важливо розподіляти витрати справедливо.

Об'єднання ресурсів. Децентралізований підхід веде до об'єднання ресурсів. Кожен вузол в системі P2P приносить певні ресурси як наприклад обчислювальна потужність або пам'ять. У програмах, які потребують величезну кількість цих ресурсів, як наприклад intensive моделювання або розподілені файлові системи, природно використовувати P2P, щоб залучити ці ресурси. Розподілені обчислювальні системи, як наприклад SETI@Home, distributed.net, і Endeavours – очевидні приклади цього підходу. Об'єднуючи ресурси тисяч вузлів, вони можуть виконувати важкі з точки зору кількості обчислень функції. Файлобмінні системи, як наприклад Napster, Gnutella, і так далі, також об'єднують

ресурси. У цих випадках, це дисковий простір, щоб зберігати дані, та пропускна спроможність, щоб їх передавати.

Вдосконалена масштабованість/надійність. З відсутністю сильної центральної влади по відношенню до автономних вузлів, важливою метою є покращення масштабованості і надійності. Масштабованість і надійність визначаються в традиційному для розподілених систем сенсі, як наприклад використання пропускної спроможності – скільки вузлів можуть бути досягнуті від одного вузла, скільки вузлів може підтримуватися, скільки користувачів може підтримуватися.

Розподілена природа peer-to-peer мереж також збільшує помилкостійкість у разі невдач, шляхом дублювання даних поміж багатьох вузлів, і – в чистих системах P2P – надаючи можливість вузлу знайти дані без залежності від єдиного централізованого індексного сервера. У останньому випадку, немає ніякої єдиної критичної точки в системі.

Збільшена автономія. У багатьох випадках, користувачі розподіленої системи не бажають залежати від будь-якого централізованого постачальника послуг. Натомість, вони воліють, щоб всі дані та призначена для них робота виконувалась локально. Системи P2P підтримують цей рівень автономії, тому що вони вимагають, щоб кожен вузол робив необхідну для нього частину праці. Анонімність/конфіденційність. Пов'язаним із автономією є поняття анонімності і конфіденційності. Користувач, можливо, не хоче, щоб кого-небудь або будь-який постачальник послуг знав про нього або про його роль у системі. З центральним сервером, гарантувати анонімність важко, тому що сервер зазвичай зможе ідентифікувати клієнта, як мінімум через його адресу в Інтернет. Використовуючи структуру P2P, в якій дії виконуються локально, користувачі можуть уникати необхідності передавати будь-яку інформацію про себе до кого-небудь іншого. FreeNet – яскравий приклад того, як анонімність може вбудуватися в додаток P2P. Він пересилає повідомлення через інші вузли, щоб забезпечити неможливість вистежування початкового автора. Це збільшує анонімність, використовуючи

ймовірнісні алгоритми таким чином, щоб походження не можливо було легко відстежити аналізуючи трафік у мережі. Динамічність. Системи P2P припускають, що оточення надзвичайно динамічне. Тобто, ресурси, як наприклад вузли, з'являються та зникають із системи безперервно. У випадках комунікації, як наприклад мережі для обміну повідомленнями, використовуються так званий «список контактів», щоб інформувати користувачів, коли їхні друзі стають доступними. Без цього, потрібно було би, щоб користувачі «опитували» партнерів, посилаючи періодичні повідомлення. У випадку розподілених обчислень, як наприклад distributed.net і SETI@home, система повинна пристосуватись до заміни учасників. Тому вони повинні повторно видавати завдання для обчислення іншим учасникам, щоб гарантувати, що робота не втрачена, якщо попередні учасники відпадають від мережі, поки вони виконували крок обчислення.

Класифікація P2P систем

За функціями:

1. Розподілені обчислення. Обчислювальна проблема розподіляються на невеликі незалежні частини. Обробка кожної з частин робиться на індивідуальному ПК і результати збираються на центральному сервері. Цей центральний сервер відповідальний за розподілення елементів роботи серед окремих комп'ютерів в Інтернеті. Кожен із зареєстрованих користувачів має клієнтське програмне забезпечення. Воно користується періодами бездіяльності в ПК (часто це характеризується часами активації скрінсейверів), щоб виконувати деяке обчислення, надане сервером.

Після того, як обчислення закінчене, результат посилається назад до сервера, і нова робота передається для клієнта.

2. Файлообмін. Зберігання та обмін даними – це одна з областей, де технологія P2P була найуспішнішою. Мультимедійні дані, наприклад, вимагають великих файлів. Napster і Gnutella використовувались користувачами, щоб обійти

обмеження пропускної спроможності, які роблять передачу великих файлів неприйнятними.

3. Співпраця. Природа технології P2P робить її добре придатною для забезпечення співпраці між користувачами. Це може бути обмін повідомленнями, онлайн ігри, сумісна робота над документами в бізнесі, освіті та дома.

За ступенем централізації:

1. Чисті peer-to-peer системи. Вузли є рівними, поєднуючи ролі серверу та клієнту. Не існує центрального сервера, що керує мережею. Прикладами таких систем є Gnutella та Freenet

2. Гібридні peer-to-peer системи. Мають центральний сервер, що зберігає інформацію про вузли та відповідає на запити відносно цієї інформації. Вузли займаються забезпеченням ресурсами (тому що центральний сервер їх не має), повідомленням сервера про наявність цих ресурсів надання ресурсів іншим вузлам, які бажають ними скористатися. В залежності від того, як вузли з'єднуються один з одним можна поділити мережі на структуровані та неструктуровані:

1. Неструктурована мережа P2P формується, коли з'єднання встановлюються довільно. Такі мережі можуть бути легко сконструйовані, оскільки новий вузол, який хоче приєднатися до мережі, може скопіювати існуючі з'єднання іншого вузла, а вже потім почати формувати свої власні. У неструктурованій мережі P2P, якщо вузол бажає знайти певні дані в мережі, запит доведеться передати майже через всю мережу, щоб охопити так багато вузлів, як можливо. Головним недоліком таких мереж є те, що запити, можливо, не завжди вирішуються. Скоріш за все популярні дані будуть доступні в багатьох вузлів та пошук швидко знайде потрібне, але якщо вузол шукає рідкісні дані, наявні лише в декількох інших вузлів, то надзвичайно малоймовірно, що пошук буде успішним. Оскільки немає ніякої кореляції між вузлами та даними, що вони зберігають, немає ніякої гарантії, що запит знайде вузол, який має бажані дані.

2. Структурована мережа P2P використовує єдиний алгоритм, щоб

гарантувати, що будь-який вузол може ефективно передати запит іншому вузлу, який має бажаний файл, навіть якщо файл надзвичайно рідкісний. Така гарантія потребує структуровану систему з'єднань. У наш час найпопулярнішим типом структурованої мережі P2P є розподілені хеш-таблиці, в яких хешування використовується для встановлення зв'язку між даними та конкретним вузлом.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Sinople – симетричний блоковий криптоалгоритм, побудований на основі незбалансованої «мережі Фейстеля». Алгоритм розроблено у 2003 році.

Основні вимоги до алгоритму при його розробці:

– Можливість програмної і апаратної реалізації.
– Висока швидкість.
– Простота.
– Низькі вимоги до пам'яті.
– Високий рівень безпеки.

Алгоритм заснований на 32-розрядних операціях і має 64 раунду, серед яких два типи – C і D. D раунди спроектовані для досягнення максимальної дифузії, C раунди – для досягнення перемішування. F-функція D раунду використовує один з елементів блоку даних (D[3]) та поточного з'єднання (K[r]) для трансформації 3-х елементів блоку даних. F-функція C раунду, навпаки, використовує перші три елемента блоку даних і поточний з'єднання (K[r]) для трансформації останнього елемента блоку даних (D[3]). Раунди D-типу виконуються до раундів C-типу. Додавання ключів з даними проводиться тільки через таблиці замін. Операції XOR (додавання за модулем 2) обов'язково поєднуються з операціями ADD (додавання за модулем 2^{32}).

Таблиці замін спочатку запозичені з алгоритму MARS і містять 512 32-розрядних елементів, проте були жорстко проаналізовано на предмет посилення.

Ключове розклад було спроектовано з урахуванням вимог:

– Простота

– Використовується та ж процедура, що і при шифруванні та розшифрування

– Установка ключа займає менше часу, ніж зашифрування

– Виключення еквівалентних ключів

– Виключення слабких ключів

Алгоритм, згідно із заявою авторів, стійкий до лінійного і диференціального аналізу.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ інтелектуальних сервісів протидії зловмисному впливу на мережу яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Функції представлені у графічному вигляді.

– Розділу обрання групи.

– Розділу виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

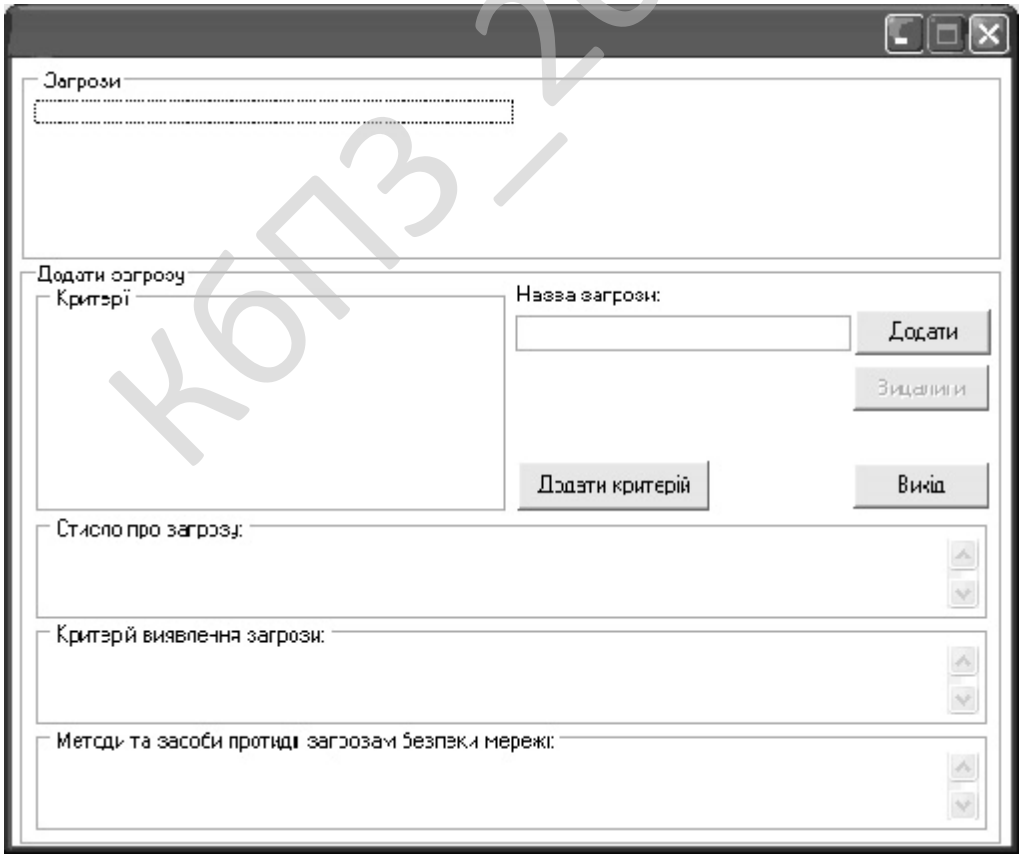


Рисунок 5.1 – Головне вікно ПЗ

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.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

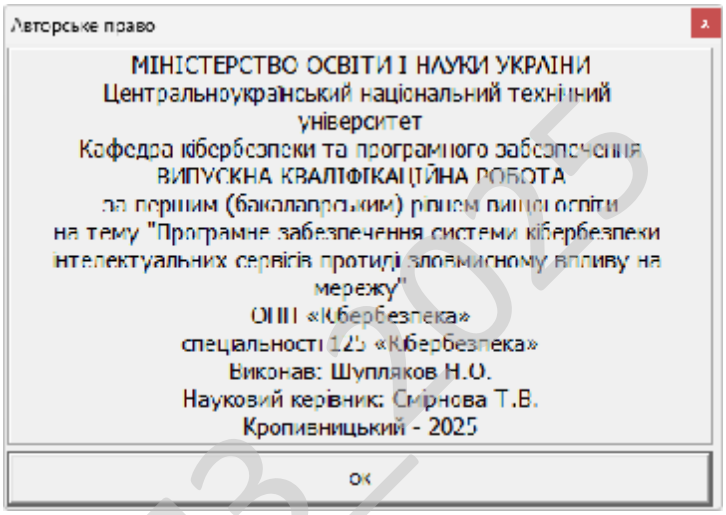


Рисунок 5.2 – Авторське право

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 ІТ 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.

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

Проводилось тестування чорної скриньки.

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

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

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

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

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

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

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

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

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

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware.

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

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

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

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

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем інтелектуальних сервісів протидії зловмисному впливу на мережу.

– Досліджена система інтелектуальних сервісів протидії зловмисному впливу на мережу.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання інтелектуальних сервісів протидії зловмисному впливу на мережу.

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 кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на

мережу. 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 кібербезпеки 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 Sinople.

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. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024, 3909, pp. 227–241.

2. 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.

3. 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.

4. 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.

5. 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.

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

7. 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

8. 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.

9. 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.

10. 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,

11. 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.

12. 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.

13. 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

14. 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.

15. 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.

16. 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.*

17. 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.

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

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

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

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special

Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

44. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

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

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.

51. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

52. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

53. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

54. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

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. № 51-02 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 кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу.

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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу;

– 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 – 101 аркуш.

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 7.06.2025 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

_____ Смірнова Т.В.

Програмне забезпечення системи кібербезпеки інтелектуальних сервісів протидії зловмисному впливу на мережу

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 21

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
#Головна система кібербезпеки інтелектуальних сервісів
# протидії зловмисному впливу на мережу

#Імпорт необхідних бібліотек
import time
import threading
import random
import datetime
import logging
import json
import hashlib
import socket
import os

#Налаштування логування
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s')

#Глобальні налаштування системи
CONFIG = {
'scan_interval': 5,
'analyze_interval': 7,
'alert_interval': 10,
'max_block_list': 100,
'simulation_mode': True,
}

#Клас для роботи з базою даних загроз
class ThreatDatabase:
    def __init__(self):
        self.threat_signatures = []
        self.load_signatures()
#Метод завантаження сигнатур загроз
    def load_signatures(self):
        self.threat_signatures = ["malware", "ddos", "phishing", "ransomware"]
        for i in range(10):
            dummy_signature = "signature" + str(i)
            self.threat_signatures.append(dummy_signature)
        return
    #Метод виконання запиту до бази загроз
    def query_threats(self, query):
        result = [sig for sig in self.threat_signatures if query in sig]
        return result
    #Метод оновлення сигнатур загроз
    def update_signatures(self, new_signatures):
        for sig in new_signatures:
            if sig not in self.threat_signatures:
                self.threat_signatures.append(sig)
        return

#Клас для сканування мережі
class NetworkScanner:
    def __init__(self):
        self.scanned_devices = []
    #Метод сканування мережі
    def scan_network(self):
        devices = []
        for i in range(1, 255, random.randint(1, 10)):
            ip = f"192.168.1.{i}"
            status = random.choice(["active", "inactive", "suspicious"])
```

```
            devices.append({'ip': ip, 'status': status})
        self.scanned_devices = devices
        return devices



#Клас для аналізу мережевого трафіку
class TrafficAnalyzer:
    def __init__(self):
        self.traffic_logs = []
#Метод аналізу трафіку
    def analyze_traffic(self):
        analysis_results = []
        for i in range(random.randint(5, 15)):
            packet = {
'source_ip': f"192.168.1.{random.randint(1,254)}",
'dest_ip': f"192.168.1.{random.randint(1,254)}",
'protocol': random.choice(["TCP", "UDP", "ICMP"]),
'data_volume': random.randint(100, 10000),
'timestamp': datetime.datetime.now().isoformat()
            }
            analysis_results.append(packet)
            self.traffic_logs.append(packet)
        return analysis_results

#Клас для виявлення вторгнень
class IntrusionDetector:
    def __init__(self):
        self.intrusion_events = []
    #Метод виявлення вторгнень на основі аналізу трафіку та бази загроз
    def detect_intrusion(self, traffic_data, threat_db):
        detected_events = []
        for packet in traffic_data:
            for threat in threat_db.threat_signatures:
                if threat in packet.get('protocol', ""):
                    event = {
'ip': packet['source_ip'],
'threat': threat,
'timestamp': packet['timestamp']
                    }
                    detected_events.append(event)
                    self.intrusion_events.append(event)
        return detected_events

#Клас для застосування заходів протидії
class CounterMeasureManager:
    def __init__(self):
        self.block_list = []
    #Метод застосування заходів протидії: блокування IP-адрес
    def apply_countermeasures(self, intrusion_events):
        actions_taken = []
        for event in intrusion_events:
            ip = event.get('ip')
            if ip and ip not in self.block_list:
                self.block_list.append(ip)
                action = f"Blocked IP: {ip}"
                actions_taken.append(action)
        return actions_taken
    #Метод отримання списку заблокованих IP
    def get_block_list(self):
        return self.block_list

#Клас для керування журналом подій
```

```
class LogManager:
    def __init__(self):
        self.logs = []
        self.log_file = "cybersecurity_logs.json"
        self.initialize_log_file()
    #Метод ініціалізації файлу журналу
    def initialize_log_file(self):
        if not os.path.exists(self.log_file):
            with open(self.log_file, "w") as f:
                json.dump([], f)
        return

    #Метод запису події в журнал
    def log_event(self, event):
        self.logs.append(event)
        self.write_logs_to_file()
        return

    #Метод запису журналу подій у файл
    def write_logs_to_file(self):
        try:
            with open(self.log_file, "w") as f:
                json.dump(self.logs, f, indent=4)
        except Exception as e:
            logging.error("Error writing logs to file: " + str(e))
        return

#Утилітна функція для перевірки цілісності даних через хешування
def validate_hash(data):
    json_data = json.dumps(data, sort_keys=True).encode()
    hash_object = hashlib.sha256(json_data)
    return hash_object.hexdigest()

#Утилітна функція для симуляції запиту до функції
def simulate_query(query_function, query_param):
    result = query_function(query_param)
    return result

#Утилітна функція для симуляції атаки
def simulate_attack(log_manager):
    event = {
'event': 'Simulated Attack',
'description': 'A simulated attack has been detected on the network.',
'timestamp': datetime.datetime.now().isoformat()
    }
    log_manager.log_event(event)
    return event

#Головний клас системи кібербезпеки
class CyberSecuritySystem:
    def __init__(self):
        self.log_manager = LogManager()
        self.network_scanner = NetworkScanner()
        self.traffic_analyzer = TrafficAnalyzer()
        self.intrusion_detector = IntrusionDetector()
        self.counter_manager = CounterMeasureManager()
        self.threat_database = ThreatDatabase()
        self.active = True
        self.query_count = 0

#Метод виконання сканування мережі та логування події
    def perform_network_scan(self):
        scanned_devices = self.network_scanner.scan_network()
```

```
        event = {
'event': 'Network Scan',
'devices_found': len(scanned_devices),
'timestamp': datetime.datetime.now().isoformat()
        }
        self.log_manager.log_event(event)
        return scanned_devices

#Метод виконання аналізу трафіку та логування події
    def perform_traffic_analysis(self):
        traffic_data = self.traffic_analyzer.analyze_traffic()
        event = {
'event': 'Traffic Analysis',
'packets_analyzed': len(traffic_data),
'timestamp': datetime.datetime.now().isoformat()
        }
        self.log_manager.log_event(event)
        return traffic_data

#Метод виявлення вторгнень і логування події
    def perform_intrusion_detection(self, traffic_data):
        detected_intrusions =
self.intrusion_detector.detect_intrusion(traffic_data, self.threat_database)
        if detected_intrusions:
            event = {
'event': 'Intrusion Detected',
'intrusions': detected_intrusions,
'timestamp': datetime.datetime.now().isoformat()
            }
            self.log_manager.log_event(event)
        return detected_intrusions

#Метод застосування заходів протидії та логування дій
    def perform_countermeasures(self, intrusion_events):
        actions = self.counter_manager.apply_countermeasures(intrusion_events)
        if actions:
            event = {
'event': 'Countermeasures Applied',
'actions': actions,
'timestamp': datetime.datetime.now().isoformat()
            }
            self.log_manager.log_event(event)
        return actions

#Метод виконання повного циклу безпеки: сканування, аналіз, виявлення, протидія
    def run_security_cycle(self):
        scanned_devices = self.perform_network_scan()
        traffic_data = self.perform_traffic_analysis()
        detected_intrusions = self.perform_intrusion_detection(traffic_data)
        if detected_intrusions:
            self.perform_countermeasures(detected_intrusions)
        current_hash = validate_hash(self.log_manager.logs)
        hash_event = {
'event': 'Hash Validation',
'hash': current_hash,
'timestamp': datetime.datetime.now().isoformat()
        }
        self.log_manager.log_event(hash_event)
        self.query_count += 1
        self.perform_additional_queries()
        return

    #Метод виконання додаткових запитів до бази загроз
```

```
    def perform_additional_queries(self):
        query_list = ["malware", "ddos", "spyware", "trojan"]
        for query in query_list:
            result = simulate_query(self.threat_database.query_threats, query)
            query_event = {
'event': 'Threat Query',
'query': query,
'result_count': len(result),
'timestamp': datetime.datetime.now().isoformat()
            }
            self.log_manager.log_event(query_event)
            self.query_count += 1
        return

    #Метод запуску системи кібербезпеки: періодичне виконання циклів безпеки
    def start_system(self):
        cycle_number = 0
        while self.active:
            cycle_number += 1
            self.run_security_cycle()
            if cycle_number % 5 == 0:
                simulate_attack(self.log_manager)
            time.sleep(CONFIG['scan_interval'])
        return
#Метод зупинки системи кібербезпеки
    def stop_system(self):
        self.active = False
        stop_event = {
'event': 'System Shutdown',
'timestamp': datetime.datetime.now().isoformat()
        }
        self.log_manager.log_event(stop_event)
        return

#Потокова функція для виконання циклів безпеки
def thread_security_cycle(system):
    while system.active:
        system.run_security_cycle()
        time.sleep(CONFIG['analyze_interval'])
    return

#Потокова функція для симуляції атаки
def thread_attack_simulation(system):
    while system.active:
        simulate_attack(system.log_manager)
        time.sleep(CONFIG['alert_interval'])
    return

#Потокова функція для виведення звіту про стан системи
def thread_status_report(system):
    while system.active:
        block_list = system.counter_manager.get_block_list()
        report = f"Status Report - Query Count: {system.query_count}, Blocked
IPs: {len(block_list)}"
        logging.info(report)
        time.sleep(15)
    return

#Головна функція запуску системи та потоків
def main():
    security_system = CyberSecuritySystem()
    cycle_thread = threading.Thread(target=thread_security_cycle,
args=(security_system,))
```

```
    attack_thread = threading.Thread(target=thread_attack_simulation,
args=(security_system,))
    report_thread = threading.Thread(target=thread_status_report,
args=(security_system,))
    cycle_thread.start()
    attack_thread.start()
    report_thread.start()
    runtime = 60
    start_time = time.time()
    while time.time() - start_time < runtime:
        time.sleep(1)
    security_system.stop_system()
    cycle_thread.join()
    attack_thread.join()
    report_thread.join()
    final_event = {
'event': 'System Terminated',
'timestamp': datetime.datetime.now().isoformat()
    }
    security_system.log_manager.log_event(final_event)
    logging.info("Cybersecurity system has been terminated.")
    return

#Додаткова функція для розширення кількості рядків коду (функція 1)
def extra_function_one():
    for i in range(50):
        dummy_value = i * random.random()
        dummy_value = dummy_value + 1
        dummy_value = dummy_value - 0.5
        if dummy_value > 10:
            dummy_value = dummy_value / 2
        else:
            dummy_value = dummy_value * 2
    return

#Додаткова функція для розширення кількості рядків коду (функція 2)
def extra_function_two():
    data_list = []
    for i in range(100):
        entry = {"index": i, "value": random.randint(0, 1000)}
        data_list.append(entry)
    sorted_list = sorted(data_list, key=lambda x: x["value"])
    for entry in sorted_list:
        temp = entry["value"] * 2
        temp = temp / 3.0
    return data_list

#Додаткова функція для розширення кількості рядків коду (функція 3)
def extra_function_three():
    result = 0
    for i in range(200):
        result += i
        result -= random.randint(0, 10)
    return result

#Додаткова функція для розширення кількості рядків коду (функція 4)
def extra_function_four():
    total = 0
    for i in range(10):
        for j in range(10):
            for k in range(10):
                total += i + j + k
    return total
```

```
#Додаткова функція для розширення кількості рядків коду (функція 5)
def extra_function_five():
    calc_result = 1
    for i in range(1, 50):
        calc_result *= i
        calc_result = calc_result % (i + 1)
    return calc_result

#Функція виклику додаткових функцій для збільшення обсягу коду
def call_extra_functions():
    extra_function_one()
    extra_function_two()
    extra_function_three()
    extra_function_four()
    extra_function_five()
    for i in range(100):
        dummy = i ** 2
        dummy = dummy + i
        dummy = dummy - i
    return

#Запуск викликів додаткових функцій
call_extra_functions()

#Основна точка входу в програму
if __name__ == '__main__':
    main()
```

Файл MultiFactorAuth.py

```
import time
import random
import threading
import datetime
import json
import hashlib
import logging
import numpy as np
from sklearn.ensemble import RandomForestClassifier

class MultiFactorAuth:
    def __init__(self):
        self.users = {}
        self.otps = {}
    def add_user(self, username, password):
        self.users[username] = hashlib.sha256(password.encode()).hexdigest()
    def generate_otp(self, username):
        otp = str(random.randint(100000, 999999))
        self.otps[username] = otp
        return otp
    def verify_password(self, username, password):
        if username in self.users:
            return self.users[username] ==
hashlib.sha256(password.encode()).hexdigest()
        return False
    def verify_otp(self, username, otp):
        if username in self.otps:
            return self.otps[username] == otp
        return False
    def login(self, username, password, otp):
        return self.verify_password(username, password) and
self.verify_otp(username, otp)

class RealTimeMonitor:
    def __init__(self):
        self.anomaly_threshold = 0.8
        self.running = True
    def monitor_traffic(self):
        while self.running:
            packet_score = random.random()
            packet_info = {
                "timestamp": datetime.datetime.now().isoformat(),
                "packet_score": packet_score,
                "source_ip": f"10.0.0.{random.randint(1,254)}",
                "dest_ip": f"10.0.1.{random.randint(1,254)}",
                "protocol": random.choice(["TCP", "UDP", "ICMP"])
            }
            if packet_score > self.anomaly_threshold:
                self.handle_anomaly(packet_info)
            time.sleep(0.5)
    def handle_anomaly(self, packet):
        anomaly_record = {
            "anomaly_detected": True,
            "details": packet,
            "detected_at": datetime.datetime.now().isoformat()
        }
        with open("anomalies.log", "a") as f:
            f.write(json.dumps(anomaly_record) + "\n")

class SIEMIntegration:
    def __init__(self):
```

```
        self.siem_logs = []
    def send_log(self, log):
        formatted_log = self.format_log(log)
        self.siem_logs.append(formatted_log)
        time.sleep(0.1)
    def format_log(self, log):
        return json.dumps(log)
    def batch_send(self, logs):
        for log in logs:
            self.send_log(log)
    def get_siem_logs(self):
        return self.siem_logs

class ThreatPredictor:
    def __init__(self):
        self.model = RandomForestClassifier(n_estimators=10, random_state=42)
        self.is_trained = False
    def train_model(self):
        X = np.random.rand(100, 5)
        y = np.random.randint(0, 2, 100)
        self.model.fit(X, y)
        self.is_trained = True
    def predict_threat(self, features):
        if not self.is_trained:
            self.train_model()
        features_array = np.array(features).reshape(1, -1)
        prediction = self.model.predict(features_array)
        return prediction[0]

class UserBehaviorAnalyzer:
    def __init__(self):
        self.user_events = {}
        self.anomaly_threshold = 5
    def add_event(self, username, event_type):
        timestamp = datetime.datetime.now().isoformat()
        event = {"event_type": event_type, "timestamp": timestamp}
        if username not in self.user_events:
            self.user_events[username] = []
        self.user_events[username].append(event)
    def analyze_behavior(self):
        anomalies = {}
        for user, events in self.user_events.items():
            if len(events) > self.anomaly_threshold:
                anomalies[user] = len(events)
        return anomalies

def main():
    auth_system = MultiFactorAuth()
    monitor_system = RealTimeMonitor()
    siem_system = SIEMIntegration()
    threat_predictor = ThreatPredictor()
    behavior_analyzer = UserBehaviorAnalyzer()
    auth_system.add_user("user1", "password1")
    auth_system.add_user("user2", "password2")
    otp_user1 = auth_system.generate_otp("user1")
    otp_user2 = auth_system.generate_otp("user2")
    login1 = auth_system.login("user1", "password1", otp_user1)
    login2 = auth_system.login("user2", "password2", otp_user2)
    def monitor_thread():
        monitor_system.monitor_traffic()
    def siem_thread():
        while True:
```

```
            log = {"timestamp": datetime.datetime.now().isoformat(), "event":
"Periodic SIEM log", "data": random.random()}
            siem_system.send_log(log)
            time.sleep(1)
    def threat_prediction_thread():
        while True:
            features = [random.random() for _ in range(5)]
            result = threat_predictor.predict_threat(features)
            with open("threat_predictions.log", "a") as f:
                f.write(json.dumps({"timestamp":
datetime.datetime.now().isoformat(), "features": features, "prediction":
int(result)}) + "\n")
            time.sleep(2)
    def behavior_thread():
        users = ["user1", "user2", "user3", "user4"]
        while True:
            user = random.choice(users)
            event_type = random.choice(["login", "file_access", "config_change",
"logout", "permission_change"])
            behavior_analyzer.add_event(user, event_type)
            anomalies = behavior_analyzer.analyze_behavior()
            with open("user_behavior.log", "a") as f:
                f.write(json.dumps({"timestamp":
datetime.datetime.now().isoformat(), "anomalies": anomalies}) + "\n")
            time.sleep(1.5)
    t1 = threading.Thread(target=monitor_thread)
    t2 = threading.Thread(target=siem_thread)
    t3 = threading.Thread(target=threat_prediction_thread)
    t4 = threading.Thread(target=behavior_thread)
    t1.daemon = True
    t2.daemon = True
    t3.daemon = True
    t4.daemon = True
    t1.start()
    t2.start()
    t3.start()
    t4.start()
    start_time = time.time()
    while time.time() - start_time < 30:
        time.sleep(0.5)
    monitor_system.running = False
    time.sleep(2)

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

Файл SignatureUpdater.py

```
import time
import random
import threading
import datetime
import json
import os
import hashlib
import random
import threading
import datetime
import json
import os

class SignatureUpdater:
    def __init__(self, threat_db):
        self.threat_db = threat_db
        self.remote_signatures = []
    def fetch_remote_signatures(self):
        self.remote_signatures = []
        for i in range(20):
            sig = "remote_sig_" + str(random.randint(1000, 9999))
            self.remote_signatures.append(sig)
        return self.remote_signatures
    def update_signatures(self):
        new_signatures = self.fetch_remote_signatures()
        self.threat_db.update_signatures(new_signatures)
    def schedule_updates(self, interval, duration):
        start_time = time.time()
        while time.time() - start_time < duration:
            self.update_signatures()
            time.sleep(interval)

class FirewallManager:
    def __init__(self):
        self.rules = []
    def add_rule(self, rule):
        self.rules.append(rule)
    def remove_rule(self, rule):
        if rule in self.rules:
            self.rules.remove(rule)
    def list_rules(self):
        return self.rules
    def apply_rules(self):
        applied_rules = []
        for rule in self.rules:
            applied_rules.append("Applied: " + rule)
        return applied_rules
    def simulate_rule_changes(self, interval, count):
        for _ in range(count):
            rule = "rule_" + str(random.randint(1, 100))
            self.add_rule(rule)
            time.sleep(interval)
        return self.list_rules()

class IncidentManager:
    def __init__(self):
        self.incidents = []
        self.incident_id = 0
    def log_incident(self, incident_type, description):
        self.incident_id += 1
```

```
        incident = {"id": self.incident_id, "type": incident_type,
"description": description, "timestamp": datetime.datetime.now().isoformat(),
"status": "new"}
        self.incidents.append(incident)
        return incident
    def escalate_incident(self, incident_id):
        for inc in self.incidents:
            if inc["id"] == incident_id:
                inc["status"] = "escalated"
                inc["escalated_at"] = datetime.datetime.now().isoformat()
                return inc
        return None
    def resolve_incident(self, incident_id):
        for inc in self.incidents:
            if inc["id"] == incident_id:
                inc["status"] = "resolved"
                inc["resolved_at"] = datetime.datetime.now().isoformat()
                return inc
        return None
    def generate_report(self):
        report = {"total": len(self.incidents), "new": 0, "escalated": 0,
"resolved": 0}
        for inc in self.incidents:
            if inc["status"] == "new":
                report["new"] += 1
            elif inc["status"] == "escalated":
                report["escalated"] += 1
            elif inc["status"] == "resolved":
                report["resolved"] += 1
        return report

class ConfigBackup:
    def __init__(self, config):
        self.config = config
        self.backup_folder = "config_backups"
        if not os.path.exists(self.backup_folder):
            os.mkdir(self.backup_folder)
    def backup(self):
        timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
        backup_file = os.path.join(self.backup_folder, "config_backup_" +
timestamp + ".json")
        with open(backup_file, "w") as f:
            json.dump(self.config, f, indent=4)
        return backup_file
    def restore(self, backup_file):
        with open(backup_file, "r") as f:
            self.config = json.load(f)
        return self.config
    def schedule_backup(self, interval, duration):
        start_time = time.time()
        backups = []
        while time.time() - start_time < duration:
            backup_file = self.backup()
            backups.append(backup_file)
            time.sleep(interval)
        return backups

class SecurityPolicyManager:
    def __init__(self, policy_file="security_policies.json"):
        self.policy_file = policy_file
        self.policies = {}
        self.load_policies()
    def load_policies(self):
```

```
        if os.path.exists(self.policy_file):
            with open(self.policy_file, "r") as f:
                try:
                    self.policies = json.load(f)
                except:
                    self.policies = {}
        else:
            self.policies = {"default": {"rule": "allow_all", "settings": {}}}
            self.save_policies()
    def save_policies(self):
        with open(self.policy_file, "w") as f:
            json.dump(self.policies, f, indent=4)
    def update_policy(self, policy_name, policy_data):
        self.policies[policy_name] = policy_data
        self.save_policies()
    def delete_policy(self, policy_name):
        if policy_name in self.policies:
            del self.policies[policy_name]
            self.save_policies()
    def list_policies(self):
        return list(self.policies.keys())
    def validate_policy(self, policy_name):
        if policy_name in self.policies:
            policy = self.policies[policy_name]
            if "rule" in policy and "settings" in policy:
                return True
        return False
    def apply_policy(self, policy_name):
        if self.validate_policy(policy_name):
            applied_policy = {"policy": policy_name, "applied_at":
datetime.datetime.now().isoformat()}
            with open("applied_policies.log", "a") as f:
                f.write(json.dumps(applied_policy) + "\n")
            return applied_policy
        return None

class ThreatDatabase:
    def __init__(self):
        self.threat_signatures = []
        self.load_signatures()
    def load_signatures(self):
        self.threat_signatures = ["malware", "ddos", "phishing", "ransomware"]
        for i in range(10):
            dummy_signature = "signature" + str(i)
            self.threat_signatures.append(dummy_signature)
        return
    def update_signatures(self, new_signatures):
        for sig in new_signatures:
            if sig not in self.threat_signatures:
                self.threat_signatures.append(sig)

CONFIG = {"scan_interval": 5, "analyze_interval": 7, "alert_interval": 10,
"max_block_list": 100, "simulation_mode": True}

def main():
    threat_db = ThreatDatabase()
    signature_updater = SignatureUpdater(threat_db)
    fw_manager = FirewallManager()
    incident_manager = IncidentManager()
    config_backup = ConfigBackup(CONFIG)
    policy_manager = SecurityPolicyManager()
    t_sig = threading.Thread(target=signature_updater.schedule_updates, args=(3,
15))
```

```
    t_fw = threading.Thread(target=fw_manager.simulate_rule_changes, args=(1,
10))
    t_backup = threading.Thread(target=config_backup.schedule_backup, args=(4,
15))
    t_sig.start()
    t_fw.start()
    t_backup.start()
    for i in range(5):
        inc = incident_manager.log_incident("intrusion", "Suspicious activity
detected " + str(i))
        if i % 2 == 0:
            incident_manager.escalate_incident(inc["id"])
        else:
            incident_manager.resolve_incident(inc["id"])
        time.sleep(1)
    policies = policy_manager.list_policies()
    policy_manager.update_policy("strict", {"rule": "deny_all", "settings":
{"exceptions": ["192.168.1.1"]}})
    applied = policy_manager.apply_policy("strict")
    report = incident_manager.generate_report()
    t_sig.join()
    t_fw.join()
    t_backup.join()
    print("Threat signatures:", threat_db.threat_signatures)
    print("Firewall rules:", fw_manager.list_rules())
    print("Incidents:", incident_manager.incidents)
    print("Backup folder:", config_backup.backup_folder)
    print("Policies:", policy_manager.policies)
    print("Incident Report:", report)

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

Файл Cuckoo.py

```
import logging
import os
import time

from cuckoo.common.abstracts import Processing
from cuckoo.common.config import config
from cuckoo.common.exceptions import (
    CuckooStartupError, CuckooOperationalError
)
from cuckoo.misc import cwd

log = logging.getLogger(__name__)

try:
    import volatility.conf as conf
    import volatility.registry as registry
    import volatility.commands as commands
    import volatility.utils as utils
    import volatility.plugins.malware.devicetree as devicetree
    import volatility.plugins.malware.apihooks as apihooks
    import volatility.plugins.getsids as sidm
    import volatility.plugins.privileges as privm
    import volatility.plugins.taskmods as taskmods
    import volatility.win32.tasks as tasks
    import volatility.obj as obj
    import volatility.exceptions as exc
    import volatility.plugins.filescan as filescan
    import volatility.protos as protos

    HAVE_VOLATILITY = True

    rootlogger = logging.getLogger()
    logging.getLogger("volatility.debug").setLevel(rootlogger.level)
    logging.getLogger("volatility.obj").setLevel(rootlogger.level)
    logging.getLogger("volatility.utils").setLevel(rootlogger.level)
except ImportError as e:
    if e.message == "No module named Crypto.Hash":
        raise CuckooStartupError(
            "Could not load Volatility: the PyCrypto package is missing "
            "(install with `pip install pycrypto`)"
        )

    if e.message.startswith("No module named volatility"):
        HAVE_VOLATILITY = False
    else:
        raise
except NameError as e:
    if "distorm3" in e.message:
        raise CuckooStartupError(
            "Could not load Volatility: the distorm3 package is missing "
            "(install with `pip install distorm3`)"
        )
    raise

def s(o):
    if isinstance(o, obj.NoneObject):
        return None
    return str(o)

class VolatilityAPI(object):
    """ Volatility API interface."""
```

```
def __init__(self, memdump, osprofile):
    """@param memdump: the memdump file path
    @param osprofile: the profile (OS type)
    """
    registry.PluginImporter()
    self.memdump = memdump
    self.osprofile = osprofile
    self.config = None
    self.addr_space = None
    self.profiles = registry.get_plugin_classes(obj.Profile).keys()
    self.init_config()

def get_dtb(self):
    """Use psscan to get system dtb and apply it."""
    ps = filescan.PSScan(self.config)

    for ep in ps.calculate():
        if str(ep.ImageFileName) == "System":
            self.config.update("dtb", ep.Pcb.DirectoryTableBase)
            return True

    return False

def init_config(self):
    """Create a volatility configuration."""
    if self.config is not None and self.addr_space is not None:
        return

    if not self.osprofile:
        raise CuckooOperationalError(
            "Can't continue to process the VM memory dump if no OS "
            "profile has been defined for it. One may define its OS "
            "profile using the 'osprofile' field for the VM in its "
            "machinery configuration or set a global default using "
            "'guest_profile' in memory.conf"
        )

    if self.osprofile not in self.profiles:
        raise CuckooOperationalError(
            "The profile '%s' does not exist! Please pick one of the "
            "following profiles for your VMs: %s" %
            (self.osprofile, ", ".join(sorted(self.profiles)))
        )

    self.config = conf.ConfObject()
    self.config.optparser.set_conflict_handler("resolve")
    registry.register_global_options(self.config, commands.Command)

    base_conf = {
        "profile": self.osprofile,
        "use_old_as": None,
        "kdbg": None,
        "help": False,
        "kpcr": None,
        "tz": None,
        "pid": None,
        "output_file": None,
        "physical_offset": None,
        "conf_file": None,
        "dtb": None,
        "output": None,
        "info": None,
```

```
            "location": "file://%s" % self.memdump,
            "plugins": None,
            "debug": None,
            "cache_dtb": True,
            "filename": None,
            "cache_directory": None,
            "verbose": None,
            "write": False
        }

        for key, value in base_conf.items():
            self.config.update(key, value)

        try:
            self.addr_space = utils.load_as(self.config)
        except exc.AddrSpaceError as e:
            if self.get_dtb():
                self.addr_space = utils.load_as(self.config)
            elif "No suitable address space mapping found" in e.message:
                raise CuckooOperationalError(
                    "An incorrect OS has been specified for this machine! "
                    "Please provide the correct one or Cuckoo won't be able "
                    "to provide Volatility-based results for analyses with "
                    "this VM."
                )
            else:
                raise

        self.plugins = (
            registry.get_plugin_classes(commands.Command, lower=True)
        )

    def pslist(self):
        """Volatility pslist plugin.
        @see volatility/plugins/taskmods.py
        """
        results = []

        command = taskmods.PSList(self.config)
        for process in command.calculate():
            results.append({
                "process_name": str(process.ImageFileName),
                "process_id": int(process.UniqueProcessId),
                "parent_id": int(process.InheritedFromUniqueProcessId),
                "num_threads": str(process.ActiveThreads),
                "num_handles": s(process.ObjectTable.HandleCount),
                "session_id": s(process.SessionId),
                "create_time": str(process.CreateTime or ""),
                "exit_time": str(process.ExitTime or ""),
            })

        return dict(config={}, data=results)

    def psxview(self):
        """Volatility psxview plugin.
        @see volatility/plugins/malware/psxview.py
        """
        results = []

        command = self.plugins["psxview"](self.config)
        for offset, process, ps_sources in command.calculate():
            results.append({
                "process_name": str(process.ImageFileName),
```

```
                "process_id": int(process.UniqueProcessId),
                "pslist": str(offset in ps_sources["pslist"]),
                "psscan": str(offset in ps_sources["psscan"]),
                "thrdproc": str(offset in ps_sources["thrdproc"]),
                "pspcid": str(offset in ps_sources["pspcid"]),
                "csrss": str(offset in ps_sources["csrss"]),
                "session": str(offset in ps_sources["session"]),
                "deskthrd": str(offset in ps_sources["deskthrd"]),
            })

        return dict(config={}, data=results)

    def callbacks(self):
        """Volatility callbacks plugin.
        @see volatility/plugins/malware/callbacks.py
        """
        results = []

        command = self.plugins["callbacks"](self.config)
        for (sym, cb, detail), mods, mod_addrs in command.calculate():
            module = tasks.find_module(
                mods, mod_addrs, self.addr_space.address_mask(cb)
            )

            if module:
                module_name = module.BaseDllName or module.FullDllName
            else:
                module_name = "UNKNOWN"

            results.append({
                "type": str(sym),
                "callback": hex(int(cb)),
                "module": str(module_name),
                "details": str(detail or "-"),
            })

        return dict(config={}, data=results)

    def idt(self):
        """Volatility idt plugin.
        @see volatility/plugins/malware/idt.py
        """
        results = []

        command = self.plugins["idt"](self.config)
        for n, entry, addr, module in command.calculate():
            if module:
                module_name = str(module.BaseDllName or "")
                sect_name = command.get_section_name(module, addr)
            else:
                module_name = "UNKNOWN"
                sect_name = ""

            cpu_number = entry.obj_parent.obj_parent.ProcessorBlock.Number
            results.append({
                "cpu_number": int(cpu_number),
                "index": int(n),
                "selector": hex(int(entry.Selector)),
                "address": hex(int(addr)),
                "module": module_name,
                "section": sect_name,
            })
```

```
        return dict(config={}, data=results)

    def gdt(self):
        """Volatility gdt plugin.
        @see volatility/plugins/malware/idt.py
        """
        results = []

        command = self.plugins["gdt"](self.config)
        for n, entry in command.calculate():
            selector = n * 8
            if entry.Present:
                present = "P"
            else:
                present = "Np"
            if entry.Type == "CallGate32":
                base = entry.CallGate
                limit = 0
                granularity = "-"
            else:
                base = entry.Base
                limit = entry.Limit
                if entry.Granularity:
                    granularity = "Pg"
                else:
                    granularity = "By"

            cpu_number = entry.obj_parent.obj_parent.ProcessorBlock.Number

            results.append({
                "cpu_number": int(cpu_number),
                "selector": hex(selector),
                "base": hex(int(base)),
                "limit": hex(int(limit)),
                "type": str(entry.Type),
                "dpl": str(entry.Dpl),
                "granularity": granularity,
                "present": present,
            })

        return dict(config={}, data=results)

    def ssdt(self):
        """Volatility ssdt plugin.
        @see volatility/plugins/ssdt.py
        """
        results = []

        command = self.plugins["ssdt"](self.config)
        syscalls = self.addr_space.profile.syscalls
        bits32 = self.addr_space.profile.metadata.get(
            "memory_model", "32bit"
        ) == "32bit"

        for idx, table, n, vm, mods, mod_addrs in command.calculate():
            for i in range(n):
                if bits32:
                    syscall_addr = obj.Object(
                        "address", table + (i * 4), vm
                    ).v()
                else:
                    offset = obj.Object("long", table + (i * 4), vm).v()
```

```
                syscall_addr = table + (offset >> 4)

            try:
                syscall_name = syscalls[idx][i]
            except IndexError:
                syscall_name = "UNKNOWN"

            syscall_mod = tasks.find_module(
                mods, mod_addrs,
                self.addr_space.address_mask(syscall_addr)
            )
            if syscall_mod:
                syscall_modname = "{0}".format(syscall_mod.BaseDllName)
            else:
                syscall_modname = "UNKNOWN"

            new = {
                "index": int(idx),
                "table": "0x%x" % int(table),
                "entry": "{0:#06x}".format(idx * 0x1000 + i),
                "syscall_name": syscall_name,
                "syscall_addr": "0x%x" % int(syscall_addr),
                "syscall_modname": syscall_modname,
            }

            if bits32 and syscall_mod is not None:
                ret = apihooks.ApiHooks.check_inline(
                    va=syscall_addr, addr_space=vm,
                    mem_start=syscall_mod.DllBase,
                    mem_end=syscall_mod.DllBase + syscall_mod.SizeOfImage)

                # Could not analyze the memory.
                if ret is not None:
                    hooked, data, dest_addr = ret
                    if hooked:
                        hook_mod = tasks.find_module(
                            mods, mod_addrs, dest_addr
                        )
                        if hook_mod:
                            hook_name = "{0}".format(hook_mod.BaseDllName)
                        else:
                            hook_name = "UNKNOWN"
                        new.update({
                            "hook_dest_addr": "{0:#x}".format(dest_addr),
                            "hook_name": hook_name,
                        })
            results.append(new)
        return dict(config={}, data=results)
```