Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра»

КвРКІ. 2202135.22.02.36 ПЗ

Виконала: студентка 2 курсу, група КІ2м-22-2 Підпис Вікторія МЕЛЬНИК (Ім'я, прізвище)

Керівник канд. техн. наук, доцент (Науковий ступінь, вчене звання) Підпис Катерина БЕРЕЗЬКА (Ім'я, прізвище)

До захисту допускаю:
Зав. кафедри КІІС, д.т.н., проф.
Тетяна ГОВОРУЩЕНКО
01 05 2024 р.

Хмельницький, 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень МАГІСТР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма ОСВІТНЬО-НАУКОВА ПРОГРАМА «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Зав. кафедри КПС
Тетяна ГОВОРУЩЕНКО
"01" 09 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Мельник Вікторія Миколаївна
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра

Керівник проекту (роботи) к.т.н., доцент Березька К.М.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.01.2024р. № 1

2. Строк подання студентом проекту (роботи) на кафедру 03.05.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

аналіз відомих методів створення кіберфізичної системи для автоматизації приміщень підприємств;

архітектура кіберфізичної системи;

метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІС		

7. Дата видачі завдання «_06_»_____09_____2023 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики ДРМ з керівником	01.09.2022	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	05.12.2023	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	25.02.2024	виконано
4	Робота над розділом 2 – розробка архітектури для вирішення поставленої задачі	01.04.2024	виконано
5	Робота над тезами	05.03.2024	виконано
6	Робота над розділом 3 – розробка методу для вирішення поставленої задачі	15.04.2024	виконано
7	Робота над розділом 4 – проектування та розробка засобів для вирішення поставленої задачі, експериментальна частина	25.04.2024	виконано
8	Оформлення пояснювальної записки згідно вимог	30.04.2024	виконано
9	Попередній захист ВКР	01.05.2024	виконано
10	Захист ВКР на засіданні ЕК	До 30.05.2024	

Студент _____ Вікторія МЕЛЬНИК
Підпис Ініціали, прізвище

Керівник роботи _____ Катерина БЕРЕЗЬКА
Підпис Ініціали, прізвище

РЕФЕРАТ

Тема кваліфікаційної роботи: «Метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра»

Автор роботи: Мельник Вікторія Миколаївна

Керівник роботи: Березька К. М.

Пояснювальна записка: 75 с., 3 рисунки, 1 таблиця, 81 джерело.

ПЕРЕЛІК КЛЮЧОВИХ СЛІВ: кіберфізичні системи; операційні системи; мікроядро; автоматизація.

Об'єктом дослідження є процес створення кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

Предметом дослідження є методи створення кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

Метою кваліфікацій роботи є розробка методу створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

Для розв'язання поставлених задач використовувалися методи теорії кіберфізичних системи,комп'ютерних мереж, архітектури комп'ютерів, теорії множин.

Наукова новизна отриманих результатів:

- розроблено новий метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

У вступі подано об'єкт та предмет дослідження, мету, наукову новизну та практичну цінність роботи, а також характеристику структури роботи.

У першому розділі проведено аналіз відомих рішень щодо створення

безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем.

У другому розділі здійснено дослідження предметної області та визначено стратегію забезпечення безпеки САБ. В якості основної технології обрана альтернативна архітектура мікроядра з мінімальним функціоналом. Запропонована стратегія відображає загальносистемний контроль доступу з локальним контролем доступу на кількох пристроях. Також, гарантує наскрізну безпеку навіть у складних середовищах, де розгортаються різноманітні протоколи та застарілі пристрої. Весь зв'язок може бути зашифрований на основі узгодження пари виділених проксі-серверів і тунельований через стандартні протоколи управління.

У третьому розділі розроблено спосіб обробки повідомлень та конфігурування мікроядра. Його реалізація базується на формі розподілу можливостей кінцевих точок, що відбувається в кореневому ініціальному користувацькому процесі. Також, розроблено метод створення безпечної кіберфізичної системи для автоматизації приміщень підприємств. Він базується на архітектурі мікроядра.

У четвертому розділі здійснено проєктування зворотньої сумісності і віртуалізації при застосуванні розробленого методу..

У висновках підведено підсумки досягнення результатів з розв'язання завдань дослідження.

На основі проведених досліджень розроблена система для автоматизації приміщень підприємств.

Практична значимість отриманих результатів полягає у розробленій крипографічній системі для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра. Вона базується на технології мікроядра для забезпечення безпеки.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

IoT – Інтернет речей

KNX —відкритий протокол мережевого зв'язку для автоматизації будівель

RTOS – операційні системи реального часу

КФС – кіберфізична система

ОС - операційна система

ПЗ - програмне забезпечення

ПІД – пропорційний інтегральний похідний контролер

ПЛК – програмовані логічні контролери

САБ – система автоматизації будівель

ВСТУП

Система автоматизації будівель (САБ) – це комплексна розподілена система управління, яка широко використовується в комерційних, житлових, промислових будівлях для моніторингу та управління механічним/електричним обладнанням. У зв'язку зі зростаючим промисловим і технологічним прогресом керуючі компоненти САБ стають все більш взаємопов'язаними. Поряд з потенційними перевагами, інтеграція також вводить нові вектори атак, що значно підвищує ризики для безпеки та захисту системи управління. Історично склалося так, що САБ не має архітектури безпеки і покладається на фізичну ізоляцію та «безпеку через невідомість». Ці методи неприйнятні для технологій «розумної будівлі». Галузі необхідно переоцінити безпеку та захист поточної системи автоматизації будівель і розробити комплексне рішення, яке забезпечить цілісність, надійність і конфіденційність як на системному, так і на мережевому рівнях.

Тому, метою роботи є розробка на системному рівні забезпечення надійної обчислювальної основи для пристроїв і контролерів. Використовуючи бажані функції безпеки, такі як надійна модульна конструкція, невеликий код привілеїв і формальну перевірюваність архітектури мікроядра, потребує опису посилена безпека операційних систем з вбудованим обов'язковим контролем доступу та структурою зв'язку на основі проксі-сервера для контролерів автоматизації будівель, тобто забезпечення функціонування кіберфізичних систем. Це рішення забезпечує зв'язок із дотриманням політики та ізоляцію між критично важливими та некритичними програмами в потенційно ворожому кіберсередовищі.

Актуальність роботи полягає в необхідності розробити метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

Метою кваліфікацийої роботи є розробка методу створення безпечних

кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

Поставлена мета досягається розв'язанням таких основних завдань:

- проаналізувати відомі методи автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра;

- розробити метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра;

- реалізувати розроблений метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра;

- здійснити есперименталькі дослідження згідно розроблених рішень.

Об'єктом дослідження є процес створення кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

Предметом дослідження є методи створення кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

Для розв'язання поставлених задач використовувалися методи теорії кіберфізичних системи, комп'ютерних мереж, архітектури комп'ютерів, теорії множин.

Наукова новизна отриманих результатів:

- розроблено новий метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

На основі проведених досліджень розроблена система для автоматизації приміщень підприємств, в основі якої організовано функціонування кіберфізичної системи.

Практична значимість отриманих результатів полягає у розробленій

крипографічній системі для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра. Вона базується на технології мікроядра для забезпечення безпеки.

За темою кваліфікаційної роботи опубліковано одну публікацію [81] у Збірнику наукових праць за матеріалами XV Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2023». (Хмельницький – 2023. – С.190-192).

1 АНАЛІЗ МЕТОДІВ І ТЕХНОЛОГІЙ СТВОРЕННЯ КІБЕРФІЗИЧНИХ СИСТЕМ ДЛЯ АВТОМАТИЗАЦІЇ ПРИМІЩЕНЬ ПІДПРИЄМСТВ З ВИКОРИСТАННЯМ ОПЕРАЦІЙНИХ СИСТЕМ НА ОСНОВІ МІКРОЯДРА

1.1 Аналіз предметної області

Системи автоматизації будівель на основі кіберфічних систем [1-4] – це великомасштабні розподілені системи керування, які спрямовані на покращення автономного керування та індивідуального управління середовищем будівлі. Типовий САБ контролює кожен аспект роботи будівлі – від освітлення, опалення, вентиляції та кондиціонування повітря до пожежної безпеки, контролю доступу, відео тощо. Операції САБ безпосередньо впливають на енергоспоживання об'єктів будівлі. Тому, зростає мотивація розробляти нові технології для САБ для підвищення комфорту пасажирів і в той же час зниження енергоспоживання та експлуатаційних витрат.

Традиційно САБ складаються з декількох автономних підсистем, специфічних для конкретних застосунків. Кожна підсистема надає лише одну послугу, таку як контроль температури, вентиляція тощо. Пристрої в кожній підсистемі пов'язані між собою через мережі автоматизації будівель, які раніше були відокремлені від IT-систем у навколишньому середовищі. Управління логікою управління здійснюється за допомогою центральних серверів управління, які називаються системами управління будівлею, які з'єднуються з програмованими логічними контролерами (ПЛК) за схемою ведучий-підлеглий. Різні підсистеми об'єднуються в різні домени управління (наприклад, домен безпеки, домен безпеки) і управляються окремо.

Зі швидким комерційним поширенням кіберфізичних систем (КФС) або широко відомих технологій Інтернету речей (IoT) у промисловості, розробники та дослідники прагнули прийняти концепцію «розумних будівель», яка використовує різні датчики для кращого розуміння життєвого контексту. Це

надає підходи з підтримкою IP для керування навколишнім середовищем та з'єднує існуючі системи керування, щоб вони краще реагували на індивідуальні потреби. Будівлі зазнають трансформації, щоб краще обслуговувати клієнтів і мешканців за допомогою передової автоматизації та об'єднання в мережу. Це, в свою чергу вимагає інтеграції все більш складних обчислювальних і мережевих можливостей в САБ, надійність і стабільність яких має вирішальне значення для підтримки повсякденного життя і забезпечення як безпеки, так і для нормальної роботи, під час надзвичайних ситуацій. Ці досягнення полегшили життя мешканцям і допомогли забудовникам знайти нові способи, наприклад, зменшити споживання енергії. Для досягнення мети енергозбереження необхідна тісна інтеграція між датчиками, органами управління, Без інтелектуальної складової частини це реалізувати неможливо. Сьогодні сучасний САБ об'єднує кілька підсистем на рівні пристроїв. Ця інтеграція означає не тільки взаємозв'язок між собою існуючих систем, але й можливість дистанційного керування та збору даних. У зв'язку з цими змінами хмарні рішення машинного навчання та системи управління, доступні в Інтернеті, починають набувати широкого поширення в САБ. Хоча, такі переваги, як гнучка реакція та каскадне управління, не викликають сумнівів, як і інші сфери, які охоплюють перспективу CPS, такі як розумні мережі, розподілена робототехніка, автономні транспортні засоби та системи авіоніки, розвиток САБ як системи зараз стикається з численними перешкодами на шляху прогресу, особливо взаємозв'язком безпеки-захисту та проблемами надійності, пов'язаними із загрозою кібератак.

Інтеграція між мережею управління та IT-мережею значно збільшує поверхню атаки САБ, пропонуючи зловмисникам більше можливостей для атаки, ніж будь-коли раніше. Кібератаки не тільки збільшують споживання енергії шляхом втручання в ретельно складені плани, але й можуть змінити функціональність компонентів керування, тим самим впливаючи на загальну безпеку будівель. Відсутність належної сумісності та стандартів безпечної

архітектури створює потенційні небезпеки та загрози для контролю, особливо для критично важливих для безпеки об'єктів.

САБ вже підключені до інтернету. Існує 21 000 систем Tridium Niagara (одна з найпопулярніших платформ для управління будівлями), підключених до Інтернету [5-9]. На сьогодні в ній було виявлено атаки у понад 50 000 будівлях, які піддаються впливу Інтернету навмисно або через неправильну конфігурацію, і можуть бути публічно обшукані за допомогою таких інструментів, як Shodan [10, 11]. Крім того, САБ широко використовує застарілі низькорівневі протоколи, які відправляють дані у вигляді відкритого тексту і не мають належних механізмів аутентифікації. В роботах [12, 13] показано, як зловмисники можуть легко отримувати доступ до керуючих пакетів, модифікувати програмований логічний контролер довільно та використовувати ретельно розроблені низькорівневі дані, зібрані за допомогою ПЛК, для впровадження програмного забезпечення високого рівня. Багато комерційних готових САБ, таких як MetaSys і Niagara [14, 15], засновані на застарілих операційних системах Windows. Відкриття першої кіберзброї, Stuxnet [6], свідчить про те, що спеціально створене зловмисне програмне забезпечення може бути легко запущено проти мереж, таких як САБ, для їх виведення з ладу, щоб компоментувати установи, які займаються високою безпекою.

Нещодавні резонансні атаки продемонстрували можливі загрози та потенційні наслідки кібератак у середовищах будівель. Наприклад, у 2013 році дослідники проникли в австралійський офіс Google через свою мережу автоматизації будівель [7]. Також, відомі випадки, коли зловмисники створили перевірку концепції зловмисного програмного забезпечення, яке продемонструвало, як зловмисники можуть використовувати системи для з'єднання мереж із зовнішнім світом [8]. У той час як Міністерство енергетики США працює над тим, щоб разом з альянсами запустити нову ініціативу для досягнення самодостатніх будівель, будівельна інфраструктура інтегрується з різними внутрішніми та зовнішніми інфраструктурами [9]. Злом цих систем

також виявився тривіальним завданням. Наприклад, взимку 2016 року хакер-новачок здійснив розподілену атаку типу «відмова в обслуговуванні» на систему опалення, вентиляції та кондиціонування повітря, що призвело до втрати опалення двох будівель у Фінляндії [10]. Такі атаки не тільки можуть допомогти зловмисникам отримати контроль над САБ, але й можуть стати сходинкою для подальшого проникнення в інші критично важливі інфраструктури, такі як електроенергія, вода, транспорт, охорона здоров'я тощо. Завдання забезпечення автоматизації будівель має кілька вимірів. По-перше, будівлі неоднорідні. Будівлі проектуються для різних цілей, а тому мають різні вимоги. Наприклад, система автоматизації стадіону може бути зосереджена на тому, як керувати освітленням, температурою та посиленою вентиляцією, щоб підтримувати низьку концентрацію вуглекислого газу. З іншого боку, головною проблемою для об'єкта біоізоляції або хімічного заводу може бути мінімізація повітрообміну між різними зонами, щоб зменшити ризик перехресного забруднення. Архітектура САБ наступного покоління повинна враховувати деталі різних сценаріїв використання та розуміти відмінності вимог, що містяться в ньому. По-друге, САБ є ієрархічною структурою. Різні підієрархії мають різні погляди на систему. Те, де і як кожна підієрархія застосовує різні правила, має значний вплив на загальну безпеку, безпеку та надійність будівлі. Життєво важливо мати глобальний погляд на систему зверху вниз з відповідною абстракцією. Формалізація різних вимог безпеки та захисту та аналіз загроз з глобальної точки зору системи, а також розробка політик та відповідних механізмів безпеки для їх підтримки повинні стати ключовим кроком у розробці САБ. Також, не менш важливою є сумісність САБ. Високоінтегровані підсистеми не тільки відкрито пов'язані між собою за допомогою мереж, але й взаємодіють за допомогою фізичних особливостей, якими вони керують, наприклад, відкриті двері змінюють характер повітряних потоків. При проектуванні БАС часто неможливо повністю ізолювати одну систему управління від іншої.

Беручи до уваги великі майбутні зміни в САБ наступного покоління та

потенційні ризики, які вони спричинять, дуже важливо переосмислити та переоцінити те, як системи автоматизації будівель проектуються та організовуються разом. Для того, щоб забезпечити необхідні гарантії безпеки в розподіленому середовищі, безпека не може вважатись другорядною ідеєю, а повинна бути одним з найважливіших міркувань усього процесу проектування та реалізації. Оцінка безпеки включає в себе не тільки механізми безпечного мережевого зв'язку, аутентифікації, але також повинна включати вбудовані платформи, архітектуру системи, операційні системи тощо. Крім того, з практичних міркувань необхідно враховувати застарілі підсистеми та стандартні протоколи промислового управління в існуючих САБ.

Таким чином, напрям роботи можна узагальнити за трьома категоріями: дослідження безпеки та захищеності в системах автоматизації будівель; безпечні операційні системи для вбудованих пристроїв; безпечна віртуалізація для вбудованих систем.

Питання безпеки та захисту систем автоматизації будівель є широкою темою досліджень і, хоча все ще перебуває на ранніх стадіях, є дуже актуальною. Роботи з питань безпеки САБ дають уявлення про проблеми безпеки в САБ [11, 12]. Розглянуті загрози безпеці на кожному рівні системи та запропонувано розширення безпеки на протоколі. Значна частина роботи зосереджена на безпеці, пов'язаній із мережевим протоколом. У статті проаналізовано використання криптографічних технологій для забезпечення безпеки автоматизації будівель зв'язок через IP-мережі [14]. Цей метод дуже схожий на мережевий тунельний підхід. Однак у цій роботі підхід тунелювання мережі є частиною мережевого зв'язку на основі проксі-сервера у спробі застосувати комунікаційні політики в існуючих САБ.

Крім того, у роботі з дослідження вразливостей у системах промислової автоматизації систематично розглядалися відомі вектори атак безпеки в системах автоматизації [15]. В цій роботі надано розширений огляд вимог безпеки для застосунків розподіленого керування в САБ [15]. У ній

обговорювалися різні технології захисту безпеки, такі як статичний аналіз, формальна верифікація тощо, і пропонувалася архітектура застосунків безпечного управління на високому рівні. Однак в ній розглядається питання безпеки та захисту з точки зору системи та зосереджується на захисті критично важливих програм керування, забезпеченні дотримання глобальних політик за допомогою надійних операційних систем. Ця робота є першою спробою розробити безпечну обчислювальну основу для вбудованих пристроїв САБ наступного покоління.

Протягом багатьох років було докладено багато зусиль для розробки безпечних операційних систем. Одним з найбільш широко адаптованих є Security-Enhanced Linux (SELinux) [12]. Він був застосований в ОС Android і експлуатується в мільйонах комерційних смартфонів. SELinux забезпечує обов'язкове керування доступом шляхом спостереження та перевірки операцій процесів відповідно до політик, збережених ядром. У SELinux передбачено комплексне керування процесами та гнучкі моделі авторизації. Він використовує модуль безпеки Linux для реалізації перевірок у ядрі Linux. SELinux знаходиться у монолітному ядрі Linux. Ефективність еталонного монітора залежить від того, чи все ядро Linux, включно з драйверами пристроїв не містять помилок безпеки. Базова ОС Linux використовує монолітну архітектуру ядра. Крім того, через складність Linux, система вимагає складного представлення станів захисту, що включає велику кількість явно визначених міток типів і політик для вираження всіх необхідних відносин доступу [11]. Як наслідок, важко, якщо взагалі можливо, зрозуміти і обґрунтувати правильність правил SELinux. Запропонований підхід використовує переваги архітектури мікроядра. Модульна структура в архітектурах мікроядер зменшує ймовірність вразливості простору ядра. Крім того, вона стосується лише примітиву IPC. Це краще підходить для вбудованого системного домену. Крім того, було запропоновано багато ОС на основі мікроядер [16-18].

З розвитком апаратного забезпечення з'явилося багато пропозицій

гіпервізора на основі мікроядер. Серед таких мікровізорів виділяються NOVA, L4Re, OKL4. В основі NOVA лежить мікроядро третього покоління з 9000 рядків коду в просторі ядра [12]. Гіпервізор NOVA використовує аналогічну модель безпеки, засновану на можливостях, і реалізує VM у просторі користувача. Однак NOVA застосовує підхід емуляції інструкцій для віртуалізації, що ускладнює конструкцію VM. На даний момент NOVA доступна тільки для архітектури X86 [19]. L4Re – це ще один фреймворк операційної системи на основі мікроядер, який фокусується на плануванні в реальному часі [13]. Подібно до мікровізора seL4, L4Re вона надає ізольовані домени. L4Re підтримує як паравіртуалізацію, так і апаратну віртуалізацію з кількома платформами, включаючи X86, ARM і MIPS. Ще один мікровізор OKL4, який є попередником seL4, підтримує паравіртуалізацію і був оптимізований до рівня високої продуктивності для комерційного використання зі фреймворками VMM і драйверами [14]. Кожен з них має свої переваги, однак, жоден з них не може забезпечити такий же рівень гарантії функціональної правильності та ізоляції [20-22], як seL4.

Таким чином, проведено аналіз предметної області для дослідження. В результаті встановлено, що кіберфізичні системи, які активно використовують а автоматизації будівель, потребують розроблення засобів і методів забезпечення їх безпеки від зовнішніх атак, оскільки використовувані операційні системи не орієнтовані на виконання саме таких завдань. За основу для напряму досліджень в частині забезпечення безпеки може бути розглянуто розроблення мікроядра операційної системи.

1.2 Методи забезпечення безпеки кіберфізичних систем

Будівлі є контейнерами безлічі різних видів людської діяльності, що підтримуються різними видами технологій. Тип і корисність будівлі визначають її унікальні потреби в безпеці [23-25]. Відсутність належної сумісності та

стандартів безпечної архітектури створює потенційні небезпеки та загрози для контролю, особливо для критично важливих для безпеки об'єктів. З іншого боку, будівлі неоднорідні [24]. Різні будівлі призначені для різних цілей, тому мають різні вимоги. Наприклад, система автоматизації стадіону може бути зосереджена на тому, як керувати освітленням, температурою та посиленою вентиляцією для підтримки низької концентрації вуглекислого газу, тоді як основною проблемою для біологічної ізоляції або хімічного заводу може бути мінімізація повітрообміну між різними зонами для зменшення ризику перехресного забруднення.

Замість того, щоб покладатися на традиційні підходи до безпеки [25-27], такі як контроль периметра та постійний моніторинг і виправлення, ми вважаємо, що вбудовані контролери повинні прийняти радикально новий архітектурний підхід до безпеки та захисту [28-31], щоб основні примітиви, що підтримують ці властивості, могли бути вбудовані, а не додані пізніше. У зв'язку з вродженою динамікою навколишнього середовища будівлі контролери повинні бути в змозі гарантувати надійну безпеку в агресивних умовах, навіть якщо деякі з них не довіряють або скомпрометовані [32]. Дуже важливим першим кроком є дослідження безлічі проблем безпеки/захисту, які можуть виникнути, і належне їх моделювання, щоб кожен вбудований контролер у навколишнє середовище мав чітко визначені вимоги безпеки/безпеки, яких він повинен дотримуватися [33]. Ґрунтуючись на аналізі сценаріїв, можна побудувати модулі конкретної системи автоматизації будівлі та визначити обмеження безпеки/захисту, які повинні бути забезпечені різними рівнями системи [34], що працює на цих вбудованих контролерах. Нарешті, можна створити багаторівневу архітектуру для вбудованих платформ, щоб програми могли використовувати ці властивості для досягнення безпеки та захисту саме для своїх контекстів [35].

Тому, проєктуватимемо рішення виходячи з того, що ОС на основі мікроядра є ядром платформи вбудованих контролерів, завдяки перевагам

мікроядра.

Ядро з базовою апаратною підтримкою забезпечує такі функціональні можливості [36]: ізоляцію процесу з метою забезпечити цілісність логіки керування; планування з обмеженням у реальному часі з метою забезпечити оперативність та доступність; міжпроцесну комунікацію та синхронізацію з метою забезпечити цілісність та автентичність потоку даних; мережевий зв'язок з метою забезпечити захищений зв'язок та конфіденційність.

Крім завдань контролю повітряного потоку і тиску, існують інші завдання управління, які співіснують на одних і тих же контролерах в зоні дослідження будівель.

Розглянемо завдання контролю безпеки [37] для забезпечення дотримання політик контролю доступу. Кожна кімната самозачиняються і дозволяє доступ лише авторизованим мешканцям. Система контролю доступу в кожній зоні управляється локальним контролером безпеки і централізовано управляється сервером фізичного доступу. У режимі роботи входити та виходити, коли доступ буде заборонено не можна, доки небезпека не буде знята.

Контрольне завдання блокування для ізоляції кожної лабораторії потрібно, щоб уникнути перехресного забруднення між лабораторіями. Кожні двері в зоні захищені магнітним блокуванням, яким керує контролер блокування. Він запрограмований на те, щоб у будь-який час можна було відчинити лише одні двері, за винятком того, що під час пожежі магнітне блокування буде вимкнено для безпеки персоналу [38].

Завдання на керування режимами для координації різних режимів роботи будівлі відбувається через управління режимами та координується глобальним контролером і кімнатними контролерами [39]. Для зони передбачено режими: режим без виділення, який вказує на етап ініціалізації зони до того, як фізичне середовище досягне потрібного рівня, наприклад, температура, тиск повітря; режим роботи, який свідчить про те, що всі лабораторії готові до використання; режим пожежі, який сигналізує про пожежу; режим дезактивації, який

спрацьовує, коли відбувається процедурне порушення або фізичне середовище падає до небажаного рівня нижче порогового значення [40].

Завдання управління пожежною сигналізацією для управління пожежною інцидентом актуальне при виявленні пожежі. Місцевий диспетчер відповідає за реагування на режим пожежі та включення сигналізації.

Завдання контролю температури для управління температурою в кожній кімнаті розглядається для контролю нагрівальної спіралі, термостату та датчику температури в приміщенні.

Управління кожною зоною здійснюється відносно незалежно за допомогою глобального контролера, яким керують система управління будівлею і система енергетичного менеджменту [41]. У зоні контроль доступу для всіх дверей здійснюється контролером безпеки, яким керує сервер фізичного доступу, тоді як керування блокуванням контролюється одним контролером блокування з магнітними приводами блокування та датчиками положення дверей, які застосовуються до кожних дверей. У кожній кімнаті є локальний контролер для всіх інших завдань керування з відповідними датчиками та виконавчими механізмами. Ці контролери обмінюються даними один з одним через протокол [42].

На відміну від сценарію для систем, що потребують ізоляції, будівлі університетських кампусів, такі як бібліотеки, гуртожитки, студентські центри тощо, призначені для забезпечення легкого та комфортного публічного доступу для великих груп та проведення різних навчальних та дослідницьких заходів [43]. У будівлях кампусу ізоляція приміщень та обмеження несанкціонованого доступу не викликають великого занепокоєння. Викладачі, співробітники та студенти проводять значну частину свого неспання в університетських будівлях. Підвищення рівня безпеки та комфорту мешканців, а також енергоефективність будівель кампусу є одними з головних пріоритетів САБ.

Основною метою управління системами опалення, вентиляції та кондиціонування повітря для будівлі кампусу САБ є ефективне збільшення

повітрообміну [44]. На відміну від сценарію, де припливне повітря повністю надходить із зовнішнього повітря через фільтри, з міркувань безпеки для будівлі кампусу енергоефективніше максимально повторно використовувати циркулююче повітря. Зазвичай свіже зовнішнє повітря, що надходить у будівлю, має бути оброблене для видалення зайвої вологи та охолоджене/нагріття до заданої температури перед подачею в будівлю. Охолодження, нагрівання та осушення повітря становить значну частину загальних витрат на експлуатацію систем опалення, вентиляції та кондиціонування повітря, особливо в районах, де погода, як правило, спекотна та волога. Таким чином, зменшення потреби в обробці зовнішнього повітря шляхом очищення та повторного використання внутрішнього повітря є важливим методом енергозбереження [45].

Повторне використання рециркуляційного повітря підвищує рівень вуглекислого газу у приміщенні. Вуглекислий газ є природним компонентом повітря. Підвищена концентрація може викликати сонливість, млявість і синдром хворої будівлі. Тому підтримка комфортного рівня концентрації в будівлях кампусу, де великі групи людей постійно видихають вуглекислий газ, має вирішальне значення [46].

Для зон з високою щільністю населення ще однією проблемою є екстрена евакуація під час пожежі та задимлення. Підсистема управління системами опалення, вентиляції та кондиціонування повітря САБ відповідає за виявлення таких інцидентів, зменшення потенційної шкоди та полегшення процедур евакуації для громадської безпеки за допомогою таких заходів, як відкриття або відмикання дверей пожежних виходів, увімкнення сигналізації пожежної сигналізації та зменшення подачі свіжого повітря в зону, що горить [47].

Щоденна послідовність роботи системи опалення, вентиляції та кондиціонування повітря наступна. Спочатку, виходячи з зовнішньої температури та вологості, керівник будівлі запускає роботу в режимі обігріву або охолодження з потрібним рівнем вологості. Коли розпочнеться робочий день, в запланований час контролер переведе систему в зайнятий режим. Інші

контролери періодично перевіряють за допомогою контролеру зміну режиму. Контролер зворотного повітря стежить за вологістю, температурою, рівнем вуглекислого газу, а також за станом пожежної тривоги. Він регулює заслінку зворотного повітря і відкриває/закриває її за бажанням керівника. У зайнятому режимі контролер зовнішнього повітря постійно регулює зовнішні заслінки, посилаючись на дані датчика від контролера зворотного повітря [48].

У зайнятому режимі контролер обробляє подачу повітря. Клапан регулювання охолодженої води змійовика охолодження підтримується на певному рівні на основі зворотного зв'язку датчика та заданого значення режиму для зниження вологості з повітря, що всмоктується. Швидкість припливного вентилятора автоматично регулюється за допомогою датчика перепаду тиску зворотного зв'язку між воздуховодом і коридором. Також є повітряний фільтр, який допомагає контролювати пил. Між двома сторонами повітряного фільтра встановлено статичний датчик, який визначає, коли фільтр засмічений настільки, що його можна замінити.

Двоповерхові контролери заслінок розгорнуті в нижній частині повітропроводу. Контролери підлогових демпферів визначають різницю тиску між повітропроводом і підлоговим коридором і використовують зворотний зв'язок для управління вхідними направляючими лопатками і заслінкою. За це відповідає контролер димовидалення для відведення диму з будівлі. У зайнятому режимі контролер відведення диму зазвичай переводиться в сплячий режим.

Під час пожежі та задимлення спрацьовує система пожежної сигналізації та сповіщається контролер. Це змінює задане значення режиму з зайнятого режиму на режим відведення диму [49-54]. Потім зміна режиму поширюється на всі контролери системи опалення, вентиляції та кондиціонування повітря. Контролер зворотного повітря повністю закриє заслінку для запобігання рециркуляції диму в будівлі [55]. Припливний вентилятор у вентиляційній установці працюватиме на повну потужність. Мета полягає в тому, щоб зменшити поширення диму на всю будівлю, забезпечуючи при цьому достатню

кількість свіжого повітря, насиченого киснем, для безпеки персоналу [56-61].

У нижній частині воздуховода контролери підлогових заслінок відкриють заслінку для відведення подачі повітря. Причина цього полягає в тому, щоб зменшити постачання кисню в офісних приміщеннях, де найімовірніше станеться пожежа, і збільшити концентрацію кисню в коридорах для полегшення евакуації [62]. Тим часом контролер відведення диму відкриє заслінку і запустить витяжний вентилятор, щоб максимально витягнути дим, поки пожежна сигналізація не буде вимкнена або ручним скиданням [63].

Аналіз цих двох сценаріїв показує, що САБ це дуже складні системи [64-75], орієнтовані на дані. Існують різні аналогові, цифрові та мережеві пристрої з даними з різних датчиків [76-79]. Тому дуже легко розгубитися і загубитися в технічних деталях. З іншого боку, сумісність підсистем САБ ускладнює проектування інтегрованої САБ. Різні компоненти взаємодіють один з одним через мережу управління, а зміни у фізичному середовищі також опосередковано впливають на прийняття рішень іншими підсистемами [80]. Наприклад, зміна температури може спричинити коливання тиску повітря, тому вимірювання перепаду тиску також має враховувати температуру в приміщенні. Часто стан і успішність завдання управління, наприклад, стан повітряного фільтра або відкриття заслінки, опосередковано позначаються фізичним станом, зафіксованим датчиками. Крім того, логіка різних контрольних завдань змінюється відповідно до інституційної політики в різних ситуаціях. Помилки в логіці управління і неправильні припущення в САБ можуть спричинити серйозну загрозу безпеці. Наприклад, у механізмах виявлення дезактивації можуть бути реалізовані дві взаємосуперечливі задачі. Якщо впровадити без ретельного обмірковування всіх можливих ситуацій у сценарії, контроль САБ може призвести до того, що забруднення ніколи не буде виявлено належним чином.

Таким чином, проведено аналіз проблем та різних сценаріїв в роботі САБ. Встановлено особливості їх функціонування в частині забезпечення безпеки.

Важливою особливістю систем автоматизаці будівель правильне проєктування, набори апаратного забезпечення та безпосередньо забезпечення правильної організації функціонування таких систем.

1.3 Висновки до першого розділу

В результаті проведеного дослідження предметної області було встановлено недоліки відомих рішень і виділено їх з метою розробки рішень, проведено аналіз проблем та різних сценаріїв в роботі САБ. Встановлено особливості їх функціонування в частині забезпечення безпеки. Важливою особливістю систем автоматизаці будівель є їх правильне проєктування для забезпечення безпеки.

1.4. Постановка задачі дослідження

Поставлена мета досягається розв'язанням таких основних завдань:

- проаналізувати відомі методи автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра;
- розробити метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра;
- реалізувати розроблений метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра;
- здійснити есперементальні дослідження згідно розроблених рішень.

2 МОДЕЛЬ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ КІБЕРФІЗИЧНИХ СИСТЕМ

2.1 Дослідження предметної області

Проведемо дослідження, які відображають основну структуру, функціональність і вимоги до сучасної САБ. Спочатку розглянемо абстрактну логічну структуру мережі керування САБ, а потім проблеми безпеки на кожному логічному рівні.

Сучасна САБ передбачає широкий обсяг зв'язку між усіма видами пристроїв і контролерів через багаторівневу промислову мережу управління, яка називається мережею автоматизації будівель (МАБ). МАБ є основою систем автоматизації будівель. Стандартну корпоративну МАБ загалом можна розділити на три ієрархічні рівні. Внизу знаходиться рівень поля, де невеликі вбудовані пристрої вимірюють і контролюють фізичне середовище. Рівень поля зазвичай складається з мережевих датчиків, наприклад, датчиків руху або температури, і виконавчих механізмів, наприклад, вентилятора, замків, змійовиків охолодження. Пристрої взаємодіють один з одним за допомогою дротових або бездротових сигналів за допомогою протоколів. Середній рівень - це місце, де здійснюється управління автоматикою. Програмовані логічні контролери, блоки збору даних і диспетчерські контролери знаходяться на цьому рівні і виконують розподілені завдання управління за допомогою мережевих протоколів. Завдання моніторингу та налаштування реалізуються на рівні управління. На цьому рівні інтегруються системи управління об'єктами, системи управління енергією та хмарні треті сторони.

За допомогою МАБ пристрої ретельно координуються між собою для виконання повсякденних операцій на об'єкті. Будівлі бувають різних розмірів і функцій; Таким чином, компоненти управління в САБ дуже неоднорідні і часто надходять від різних постачальників з власними комунікаційними рішеннями. Таким комунікаційним рішенням часто не вистачає безпеки. Крім того, завдяки

тривалому життєвому циклу, БАС часто складаються з численних застарілих підсистеми. Такий різноманітний спектр засобів зв'язку та пристроїв є складним завданням для безпеки всієї системи.

Мережеві протоколи є основою САБ. Протокол зв'язку BACnet є одним з найпопулярніших протоколів в мережах автоматизації та управління будівлями. Цей протокол розроблений для управління системами опалення, вентиляції та кондиціонування повітря. Як стандарт зв'язку з відкритим протоколом широко застосовується до різних типів практичних систем у багатьох галузях, таких як опалення, вентиляція, освітлення, безпека життєдіяльності. Це стандарт автоматизації будівель завдяки своїй відкритості, гнучкості та сумісності. Він підтримує кілька різних протоколів фізичного та канального рівня. Його можна використовувати для інтеграції різних типів і постачальників без необхідності використовувати спеціальне обладнання. Протокол підтримує шаблони та сервіси зв'язку «опитування-відповідь» та «публікація-підписка», включаючи виявлення пристроїв та об'єктів. Дані всередині пристрою організовані як ряд об'єктів. Кожна частина інформації інкапсульована в об'єкті. Кожен об'єкт має тип і набір властивостей. Однак основна увага зосереджена на сумісності та функціях, але з історичної причини на практиці він не має належної криптографічної підтримки та механізму аутентифікації.

Крім нього, також є інші протоколи. Наприклад, є мережева платформа, створена для задоволення потреб керуючих застосунків. Це власний стандарт зв'язку, що включає в себе кілька протоколів сімейства, розроблених на початку 90-х років. Стандарт заснований на низькорівневому протоколі зв'язку. Як конкуруючий стандарт протоколу пропонує інший тип вирішення проблеми сумісності. Підтримує різні типи носіїв передачі, такі як вита пара, лінія електропередачі, радіочастота, оптоволокно або IP, як TCP/IP, так і UDP/IP. Він вимагає спеціалізованого комунікаційного мікропроцесора і разом з програмним забезпеченням, що його супроводжує, протокол забезпечує зв'язок та обмін інформацією між пристроями. Інший протокол визначає зміст і структуру

інформації, якою обмінюються.

Ще одним широко використовуваним відкритим протоколом в автоматизації будівель є протокол для використання в системах промислової автоматизації з програмованими контролерами. Це один з найбільш широко використовуваних засобів для підключення електронного обладнання в промислових цілях. Він підтримує традиційні послідовні протоколи та протоколи Ethernet і має прості мінімальні вимоги до апаратного забезпечення. Перевага полягає в тому, що він використовує транспортний протокол TCP/IP, отже, може бути легко інтегрований з Інтернетом. Однак сам цей протокол не забезпечує захисту від несанкціонованих команд або перехоплення даних.

Крім цих протоколів, існують і інші стандарти зв'язку в області автоматизації будівель. KNX — це відкритий протокол мережевого зв'язку для автоматизації будівель, який використовується понад 20 років і підтримується сотнями постачальників. DALI – це стандартизований інтерфейс для керування освітленням. EnOcean – це бездротова технологія збору енергії, яка використовується в основному в системах автоматизації будівель у відповідь на концепцію розумного будинку. Нерідкі випадки, коли в системі автоматизації будівлі співіснує більше одного протоколу. Різні частини підсистеми САБ, особливо надані різними постачальниками, можуть використовувати різні протоколи. Також часто можна побачити, що різні рівні системи використовують різні протоколи. Наприклад, у системі між головним і підлеглим пристроями вони можуть обмінюватися даними, але зв'язок між пристроями автоматизації та системою керування може використовувати другий протокол тощо.

На рівні мережі автоматизації будівлі знаходяться різні датчики та виконавчі механізми, наприклад, датчик температури, датчик тиску повітря, лічильник електроенергії, термостат, обігрівач, заслінка, клапан повітряного потоку тощо. Датчики та виконавчі механізми підключалися до контролерів через прямі інтерфейси або прості мережі, такі як дротові шини, які підтримують

кручену пару як мережеві носії. Датчики переводять аналогові сигнали на цифровий вхід для контролерів. Виконавчі механізми приймають вихідні керуючі сигнали для регулювання фізичного середовища.

Традиційно ПЛК відіграють важливу роль у системах керування. ПЛК – це цифровий вбудований пристрій, який використовується для автоматизації промислових процесів. Вони здійснюють управління електромеханічними процесами в режимі реального часу. ПЛК працює шляхом постійного сканування програми, що називається циклом сканування. Це передбачає багаторазове зчитування вимірювань датчика, виконання програми логіки керування для обчислення вихідної потужності та приведення в дію електромеханічних процесів. ПЛК відповідають за постійне налаштування фізичного обладнання на основі вимірювання датчиком, а також функціонують як шлюз між машиною та людьми-операторами. ПЛК перетворюють безперервні аналогові сигнали в цифрові значення. Обробка виконується за допомогою циклу введення, обробки та виведення. Логіка управління обробкою, як правило, представлена за допомогою релейної драбинчастої логіки, написаної графічною мовою. Найбільш широко використовуваною структурою управління ПЛК є пропорційний інтегральний похідний (ПІД) контролер. Цей алгоритм контролера має три окремі константні параметри: пропорційне, інтегральне та похідне. ПІД-регулятори використовуються для динамічного регулювання вихідного сигналу шляхом порівняння заданого бажаного значення і фактичного значення змінної процесу з контролюваного процесу. Однак таку логіку управління можна легко змоделювати за допомогою програмного забезпечення.

Тенденція до більш інтелектуального управління в САБ вимагає інтеграції даних і нових можливостей в пристроях управління. Для того, щоб задовольнити цю вимогу, сучасний САБ вимагає більшої обчислювальної потужності. На зміну ПЛК прийшли вбудовані контролери нового покоління, що запускають процеси з більшим функціоналом. З розвитком апаратного забезпечення багато постачальників контролерів автоматизації поступово переходять на архітектуру

як основну обчислювальну платформу поточних і майбутніх контролерів будівель. Одним із прикладів є контролер, який використовується при автоматизації будівель. Ці пристрої використовують процесор і переважно працюють під управлінням UNIX-подібної операційної системи.

Поряд з логікою управління програмним забезпеченням, хмарні клієнти, веб-сервер, аналітика часто розгортаються на одних і тих же вбудованих контролерах, що підвищує складність системи і робить її схильною до потенційних вразливостей і експлойтів. Традиційні операційні системи для вбудованих контролерів є одночасно комерційними ОС, наприклад, Linux, Windows, які охоплюють всі служби ОС у привілейованому режимі завдань у просторі ядра та розподіляють ресурси за допомогою дискреційного контролю доступу. Компрометація будь-якого коду, що працює в привілейованому режимі, дає зловмисникам найвищі привілеї в системі, таким чином обходячи будь-які засоби контролю безпеки. Для того, щоб захистити критично важливі для безпеки застосування будівель у таких системах змішаної критичності, надійна розподілена вбудована архітектура повинна ізолювати застосунки та регулювати використання локальних ресурсів, щоб навіть коли менш критичні програми були скомпрометовані зловмисниками, критичні з них залишалися безпечними у безпечний спосіб.

САБ може використовувати програмне забезпечення для моніторингу об'єктів автоматизації. SCADA часто використовується в розподілених системах віддаленого моніторингу та управління, які функціонують по мережах зв'язку. SCADA здійснює централізований моніторинг та управління об'єктами на великих відстанях, такими як електричні мережі, університетське містечко, включаючи моніторинг тривог та обробку даних про стан. На основі інформації, отриманої від віддалених пристроїв, диспетчерські команди, автоматизовані або видані операторами через людино-машинний інтерфейс, можуть бути передані на пристрої дистанційного керування. SCADA-сервери, які використовуються в САБ, часто не відрізняються від робочих станцій ІТ-комп'ютерів. Більшість

SCADA-систем, що використовують САБ, часто працюють під управлінням стандартних операційних систем Windows або Linux з програмним забезпеченням на фізичному хості або у віртуальній інфраструктурі. Деякі постачальники надають свою індивідуальну операційну систему, однак, згідно з посібником з безпеки промислових систем управління, вона часто не має вбудованих можливостей безпеки. В даний час, у зв'язку з тим, що обчислювальні потужності різко зросли і доступні за економічно вигідною ціною, більш потужні пристрої автоматизації та диспетчеризації були розгорнуті на рівні автоматизації, що забезпечує віддалений доступ через веб-інтерфейс. Типовим прикладом є рішення, яке має більшу частину функціоналу управління мережевої автоматизації і працює під управлінням вбудованої Windows. В результаті САБ починають інтегруватися з корпоративною IT-інфраструктурою і часто управляються IT-командами. Отже, контролери САБ з'єднані між собою загальною магістральною мережею TCP/IP через виділені шлюзи та VLAN.

Стандартні протоколи були розроблені в 80-х роках. Вони не підтримують сучасні механізми безпеки, і часто не враховують безпеку в своїй конструкції. У 2012 році була опублікована остання специфікація мережі, яка представила керівні принципи підтримки стандартного криптографічного захисту. Однак, нові пристрої не підтримують ці специфікації з міркувань зворотної сумісності. Як наслідок, операторам САБ доводиться покладатися на безпеку з певною ймовірністю. Нібито мережі ізольовані від зовнішнього впливу, оскільки вони не підключені до зовнішніх мереж. У зв'язку з тим, що сучасні САБ переходять до більш інтелектуальних систем управління, які значною мірою покладаються на зв'язок із зовнішнім світом, такий підхід може бути навіть нездійсненним. Після того, як повітряний прошарок буде порушено, весь простір стане вразливим для атаки.

Багато закладів покладаються на брандмауери-шлюзи для безпеки. Останніми роками також було докладено багато зусиль для розробки системи виявлення вторгнень (IDS) для промислових систем управління (ICS), які мають

багато спільного з САБ. Незважаючи на те, що вони забезпечують додатковий рівень захисту, самих лише брандмауерів та IDS недостатньо. Брандмауери повинні пропускати певний трафік, а кібератаки можуть успішно здійснюватися законними каналами зв'язку. У зв'язку з обмеженнями реального часу в більшості ІКС важко реагувати на підозрілий трафік, виявлений IDS, IDS і міжмережеві екрани необхідні для зупинки ряду атак з відомими сигнатурами, але недостатні для зупинки атак без знання семантики різних логік управління, особливо з огляду на той факт, що традиційно IDS має тенденцію виробляти велику кількість помилкових спрацьовувань. Це може бути особливо проблематично в середовищі САБ, де пристрої дуже неоднорідні. Крім того, системи з брандмауерами переважно використовують дискреційні моделі контролю доступу. Програми або повністю дозволяються, або їм відмовляють у доступі до мережі на основі ідентичності програм. Цієї моделі недостатньо, оскільки скомпрометовані застосунки з авторизованою ідентифікацією дозволять зловмисникам довільно зловживати мережевими ресурсами. Тісна інтеграція між підсистемами вимагає точного контролю доступу.

Загрози походять не лише з мережі, але й із програмних вразливостей у контролерах автоматизації. Дослідження надійності програмного забезпечення показує, що промислова програмна система, в загальному випадку, містить від багато рядків виконуваного коду, що значно перевищує нормальний код. З іншого боку, виправлення вразливостей безпеки в системах управління утруднене і зазвичай доводиться чекати при проведенні технічного обслуговування. Ці проблеми роблять системи управління, такі як САБ, привабливою мішенню для зловмисників. Системи управління є високоавтоматизованими та цифрово взаємопов'язаними з високорівневими застосунками та Інтернетом. З огляду на легку доступність, поганий захист і потенційну вигоду, яку можуть отримати хакери, контролери автоматизації стали обзєктами для кібератак. Змінилося середовище і уявлення про операційні системи реального часу для САБ. На сьогодні не можна вважати, що всі

застосунки, що працюють поверх ОС, є доброякісними. Наприклад, з новою конфігурацією програма з уразливістю переповнення буфера може дозволити зловмисникам віддалено впроваджувати шкідливий код, обманюючи планувальник, змушуючи САБ порушувати обмеження в реальному часі.

Традиційно контролери автоматизації працюють на простих операційних системах реального часу (RTOS), хоча цих RTOS достатньо, вони в основному зосереджені на функціональних можливостях збірки та вимогах до реального часу, а не на безпеці та захисті. Ці застарілі контролери часто не мають сучасних функцій ОС, таких як віртуальні модулі пам'яті та захисні кільця. Безпека ґрунтується на фізичному відокремленні, використовуючи різні мікроконтролери для кожної функції. Таке рішення не є витратним матеріалом. Існує багато мікроконтролерів для відносно невеликого сценарію. Коли розгортається все більше і більше мікроконтролерів, мережа стає складною і важкою в управлінні. Вони вимагають багато кабелів, споживають багато енергії, головне це те, що мікроконтролери коштують недешево. Ця проблема ускладнюється збільшенням кількості датчиків, які повинні бути спільними для різних пристроїв.

В даний час сучасні контролери автоматизації часто містять потужний процесор і розміщують безліч завдань управління. Вони часто працюють під управлінням комерційних операційних систем, які дуже схожі на операційні системи для комп'ютерів загального призначення, такі як вбудовані Windows, VxWorks, QNX, Linux тощо. Більшість комерційних операційних систем розробляються на основі монолітної архітектури ядра. У монолітній архітектурі всі функції ОС, включаючи обробники переривань, драйвери пристроїв і функції системних викликів, реалізовані як функції в одному великому двійковому файлі ядра, що виконується в одному адресному просторі на найвищому рівні привілеїв. Таким чином, будь-які вразливості в одній частині ядра, такі як драйвер стороннього пристрою, можуть дозволити зловмиснику виконати будь-які конфіденційні операції та отримати привілеї ядра. Крім того, монолітна

архітектура набиває всі функціональні можливості системи в одному адресному просторі, що робить код ядра роздутим. Типове ядро Linux містить понад два мільйони рядків коду. Це унеможливлює гарантування надійності найважливішого компонента контролера автоматизації, а отже, ставить під загрозу безпеку та захищеність системи автоматизації будівлі.

Ще одним важливим питанням, яке пов'язане з операційною системою контролера автоматизації, є ізоляція процесу. У більшості Unix-подібних систем міжпроцесний зв'язок (IPC) здійснюється або через черги повідомлень, або через локальні сокети Unix, які реалізуються через файлову систему. Таким чином, автентичність повідомлення захищається за допомогою дозволів на доступ до файлів і надається за допомогою дискреційного керування доступом на основі ролей. Рішення за допомогою гнучкості забезпечує досить обмежену безпеку. Якщо ці дескриптори файлової системи не налаштовані належним чином, то вони можуть бути використані зловмисниками. Якщо процес скомпрометовано зловмисниками, цілком можливо, що скомпрометований процес може використовувати IPC для взаємодії з одноранговим процесом і впроваджує повідомлення про підробку.

З іншого боку, традиційні Unix-подібні ОС створені для надання апаратних ресурсів для добросовісного використання програмами. Будь-який процес має однаковий привілей запитувати послуги у драйверів пристроїв. Щоб вести мережевий зв'язок, процеси можуть відкривати порти, просто запросивши сокет і запис до файлу, зіставленого з пам'яттю. По суті, будь-які скомпрометовані процеси можуть надсилати вихідний зв'язок і проводити DoS-атаки в мережі. Більше того, якщо шкідливий процес отримує більше привілеїв, то він може контролювати весь пристрій і довільно маніпулювати вхідним і вихідним мережевим трафіком. На відміну від обчислювальних платформ загального призначення, вбудовані пристрої, такі як контролери САБ, часто спільно розміщують застосунки зі змішаною критичністю та різними вимогами до реального часу. Побудова таких систем з використанням традиційної моделі

спільного використання ресурсів ОС створить ситуації, коли скомпрометовані процеси в САБ можуть легко зловживати мережевими ресурсами і поширювати шкідливе програмне забезпечення по всій мережі. Нещодавні масові DDoS-атаки, які використовували скомпрометовані ботнети IoT, є символічними прикладами цієї проблеми. В критично важливих для безпеки системах мережевий зв'язок повинен бути привілейованою операцією.

Система управління системою автоматизації будівель дуже схожа на IT-систему підприємства. Отже, загрози безпеці також схожі з IT-системою. По-перше, сервери часто схильні до комп'ютерних вірусів і бекдорів. Багато механізмів управління САБ працюють під управлінням вбудованої Windows через уразливості в програмному забезпеченні та системному сервісі. Системи управління мають вимогу до високої надійності, легко контролювати оновлення системи та застосовувати виправлення програмного забезпечення.

Сучасна система управління САБ часто підтримує веб-інтерфейс і доступ до неї можна отримати через браузер. Якщо система не налаштована належним чином, то вона вразлива до типових веб-атак. Наприклад, якщо сертифікат веб-сервера налаштований неправильно, то зловмисники можуть проводити прослуховування, атаки типу "людина посередині" та інші поширені інтернет-атаки. Крім того, для зручності багато системних адміністраторів налаштовують віддалений доступ в мережі управління САБ. Незважаючи на те, що такий доступ до мережі часто захищений VLAN, VPN тощо, якщо в мережах є неправильні конфігурації або адміністратори мали слабкі паролі, то можна виявити слабке місце і дозволити атакам отримати доступ. І останнє, але не менш важливе: аутентифікація між клієнтом і сервером САБ має вирішальне значення. Гучні системні злами часто починаються з компрометації пароля або через експлуатацію системи, наприклад, слабкий злом пароля, обхід автентифікації, або через експлуатацію користувачів, наприклад, фішингові атаки та інші методи соціальної інженерії.

Виходячи з вищесказаного, питання безпеки сучасних систем

автоматизації будівель можна розділити на наступні три рівні атаки:

1. Атака на систему управління.

2. Атака на внутрішню мережу та протокол.

3. Атака на контролери автоматизації та пристрої.

Загрози для адміністратора системи: бекдор; вірус; вразливості в поширених протоколах; атака "людина посередині"; атака спуфінгу мережі; відмова в обслуговуванні; підслуховування; атака на вразливості програмного забезпечення та сервісів; внутрішні загрози; соціальна інженерія.

Загрози при проєктуванні мережі автоматизації: мережевий сніфінг (перехоплення); мережева спуфінг-атака (фабрикація); вставка неправильно сформованих даних; вставка фальшивих команд; атака мережевого повтору; атака "людина посередині" (модифікація); атака типу «відмова в обслуговуванні » (переривання).

Загрози автоматизації: фізичне втручання; злом паролів; атака обману; повторна атака; видавання себе за іншу особу; мережева атака; підвищення привілеїв; ін'єкція неправдивих даних IPC; руткіти; фальшива команда; атака на монополію ресурсів; атака на споживання ресурсів; ін'єкція коду; слабкість алгоритму експлойту.

Прагнучи вирішити вищезгадані проблеми безпеки, розроблятимемо надійну обчислювальну платформу, яка дотримується принципу безпеки, щоб пом'якшити проблеми. Для того, щоб надійна обчислювальна платформа підтримувала систему змішаної критичності, архітектура зводиться до ізоляції, автентичності та безпечного зв'язку. Обчислювальну платформу можна розглядати як багаторівневу архітектуру. У традиційному підході до вбудованих систем всі рівні є частиною довірчої обчислювальної бази, а це означає, що вони мають вирішальне значення для безпеки системи. Якщо треба створити безпечну платформу, то треба переконатися, що всі ці елементи захищені. Це велика частина системи, яку потрібно убезпечити з урахуванням рядків коду.

Таким чином, здійснено дослідження предметної області та визначено стратегію забезпечення безпеки САБ. В САБ недостатньо комплексно вирішено питання забезпечення безпеки. Це пов'язано з використанням різних технологій і тим, що розробники технологічних рішень не враховували таку проблему. При дослідженні різних архітектур операційних систем було встановлено, що архітектура операційних систем, в якій в ядрі включені всі системні функції, може бути успішно атакована. Тому, в якості основної технології обрана альтернативна архітектура мікроядра з мінімальним функціоналом.

2.2 Архітектура мікроядра проєктованої системи з безпекою

У сучасному програмному забезпеченні операційні системи є ядром будь-якої обчислювальної системи. Долаючи межу між апаратним і програмним забезпеченням, ОС керують апаратними та фізичними ресурсами, такими як пам'ять, периферійні пристрої, використання процесора, енергія тощо для ізоляції та спільної роботи. З точки зору програми, ОС абстрагує складні деталі різних апаратних механізмів і надає послуги за допомогою простих у використанні інтерфейсів. Отже, безпека та стабільність системи значною мірою залежать від надійності та безпеки операційної системи. Основна частина операційної системи називається ядром. Ядро - це перша програма, що виконується в системі. Працюючи в найбільш привілейованому режимі, він відповідає за ініціалізацію платформи та початкове завантаження інших сервісів.

Взагалі кажучи, існує дві основні архітектури ядра: монолітна архітектура ядра та архітектура мікроядра. Будучи мінімалістичною системою, мікроядро має примітиви міжпроцесного зв'язку, що використовують передачу повідомлень і управління процесами, а також виділяє введення-виведення для планування зовнішніх процесів. Мікроядро стало дуже привабливим рішенням у відповідь на існуючі проблеми безпеки та надійності в монолітних операційних

системах на основі ядра.

За певний період часу з'явилося багато експериментальних мікроядер, одними з відомих є MINIX, QNX, AmigaOS, які все ще знаходяться на стадії активної розробки.

Довгий час термін «мікроядро» асоціювався з низькою продуктивністю. Однак таке сприйняття не обов'язково відповідає дійсності. Насправді, проблеми з продуктивністю не притаманні архітектурі мікроядер, а пов'язані зі складністю існуючих варіантів архітектури. Наприклад, багато мікроядер надають як синхронні, так і асинхронні IPC і підтримують понад 100 системних викликів для того, щоб бути сумісними з системою UNIX. Перше мікроядро L4 показує, що мікроядра не обов'язково повинні бути повільними. Написаний на асемблерному коді і спеціально орієнтований на високу продуктивність. Принцип мінімальності: «Концепція допускається всередині мікроядра тільки в тому випадку, якщо переміщення її за межі ядра, тобто дозвіл конкуруючих імплементацій, перешкоджає реалізації необхідної функціональності системи». Після успіху першого покоління мікроядер L4 з'явилося багато мікроядер сімейства L4 як в академічних, експериментальних, так і в комерційних розгортаннях. Ці реалізації включають не тільки L4, але й інші: концепція мікроядра, сконструйованого на високорівневій мові C++; концепція ядра призначена для підтримки жорсткого режиму реального часу та фокусуванні на досягненні низької затримки переривань; концепція ядра, яка підтримує багатопроцесорність. Мікроядро третього покоління seL4. SeL4 – це перше формально перевірене мікроядро у світі, яке орієнтоване на високу надійність. З розвитком апаратного забезпечення та зростаючою турботою про безпеку, ідея мікроядра вдруге повернулася, оскільки було запропоновано багато експериментальних ОС на базі мікроядер.

Конструкція мікроядра відповідає принципу найменших привілеїв в інформаційній безпеці. На відміну від монолітної архітектури ядра, де всі служби ОС і драйвери виконуються в адресному просторі ядра, архітектура

розроблена в модульній гнучкій структурі. У типових операційних системах на основі мікроядер ядро обробляє лише низькорівневу функціональність і примітиви процесів, такі як низькорівневе керування адресами, апаратні блоки керування процесами переривань та IPC. Всі інші системні служби ОС, такі як файлова система, управління віртуальною пам'яттю, мережевий стек і драйвери пристроїв, які традиційно знаходяться в просторі ядра, видаляються з ядра і виконуються як процеси в режимі користувача. Архітектура більш надійна в порівнянні з монолітним підходом ядра. Це викликано зменшенням розміру коду, що працює в привілейованому режимі, що різко мінімізує катастрофічні помилки. Крім того, всі процеси в режимі користувача не мають повноважень для прямого доступу до пам'яті, яка не належить цим процесам, тому вони добре ізольовані.

В архітектурі мікроядра міжпроцесний зв'язок є основою. На відміну від монолітного ядра, де всі функції системи знаходяться в найбільш привілейованому режимі процесора, різні частини системи можуть безпосередньо викликати одна одну за допомогою виклику функції. Мікроядро організовується за допомогою виклику віддаленої процедури IPC. Оскільки різні служби працюють в ізольованому просторі пам'яті, дані явно передаються або через регістри, або через спільну пам'ять.

Мотивація, яка керує вибором архітектури мікроядра, полягає в тому, як сприймається ядро. Традиційно ядро розглядалося як центральна особа, яка приймає рішення, яка має на меті управління ресурсами, приховує складність апаратного забезпечення та надає справедливі послуги якомога ефективніше, припускаючи, що кожен фрагмент коду є доброякісним. З кожною системною службою, що працює в просторі ядра, результатом є роздуте монолітне ядро. Від драйверів пристроїв до файлової системи, кожна компонента системи може містити вразливості. Сучасні операційні системи є складним програмним забезпеченням. Ядро Linux має понад 2,5 мільйона рядків коду, а системи Windows ще більші. Гарантувати функціональну коректність монолітного ядра

неможливо. Структура мікроядра пом'якшує відповідальність ядра і розглядає ядро як органайзер, подібно до функціональності комутатора мережі, який лише забезпечує механізми зв'язку для вільного зв'язку різних модулів. Така структура забезпечує наступні суттєві переваги в порівнянні з монолітною структурою ядра. По-перше, ізоляція системних служб і драйверів пристроїв значно зменшує ймовірність порушення безпеки в просторі ядра. Загальновідомо, що у драйверів пристроїв частота помилок в 3-7 разів вище, ніж в інших кодах. В ОС загального призначення драйвери пристроїв сприяють великому відсотку вразливостей. У монолітній архітектурі ядра компрометація драйверів пристроїв надасть зловмисникам найвищі привілеї, тоді як у мікроядрі драйвери пристроїв є процесами простору користувача. Якщо тільки ядро скомпрометовано, то злам добре обмежений вразливими процесами. Експлойти коду драйвера не призведуть до підвищення привілеїв, а решта функціоналу залишається недоторканою. По-друге, мікроядра більш стійкі до збоїв через зменшений розмір коду в просторі ядра. Функціональна коректність багато в чому пов'язана з розміром ядра. Типове мікроядро складається приблизно з 4000 - 6000 рядків коду. Невеликий розмір коду в мікроядрі полегшує аудит коду. Як показує аналіз seL4, можна навіть застосувати формальну верифікацію до його кодової бази і математично довести специфікацію і реалізацію. Крім того, мікроядра, як правило, динамічно не виділяють пам'ять. Це різко знижує ймовірність атак на основі пам'яті, таких як переповнення буфера та експлойти «після використання» в просторі ядра. Нарешті, унікальна модульність мікроядра гарантує ізоляцію процесу і гарантує, що існує тільки один спосіб зв'язку між різними процесами. Єдиний міжпроцесний зв'язок у системі мікроядра — це примітиви ядра, такі як передача повідомлень. З точки зору безпеки, це дозволяє розробникам легко відстежувати поведінку процесів, системні виклики тощо та виявляти потенційні експлойти. Крім того, через цю винятковість, ядро знаходиться в унікальному становищі дозволяти, заперечувати або іншим чином рецензувати будь-яку комунікацію. У порівнянні з монолітною ОС, всі функції

в ОС виконуються від імені процесу користувача і перемикання процесів не відбувається. Важко відслідковувати ці виклики функцій після того, як керування передано ядру.

В архітектурі мікроядра підсистеми та компоненти простору користувача модульовані в ізольовані процеси. Міжпроцесний зв'язок в архітектурі мікроядра подібний до зв'язку в розподілених мережах, де ядро поводиться як маршрутизатор. Для того, щоб процеси могли надсилати дані або сигнали через простір пам'яті, процеси повинні використовувати примітив ядра, передачу повідомлень. Завдяки цій ексклюзивності, ядро знаходиться в унікальному становищі дозволяти, відхиляти або іншим чином рецензувати всі IPC. З іншого боку, у вбудованих системах IPC, як правило, дотримується заздалегідь визначених шаблонів між процесами простору користувача. Регулюючи використання примітивів IPC, ядро арбітражує взаємодію між ізольованими процесами. Він може застосовувати індивідуальну політику контролю для всіх потоків даних на основі специфікації системи.

Таким чином, для забезпечення дотримання комунікаційної політики для систем змішаної критичності, ідеальним рішенням є розробка механізму контролю доступу для регулювання IPC в архітектурі мікроядра. Саме такий напрям дослідження є перспективним і потребує розроблення.

Поширеним примітивом IPC в архітектурах мікроядер є синхронна передача повідомлень. Використовуючи системний виклик, процес запитує ядро надіслати або отримати повідомлення. На цьому етапі ядро може визначити обґрунтованість і автентичність цього запиту на основі заздалегідь визначених політик IPC. Після того, як повідомлення перевірено, ядро передає повідомлення через межі адресного простору через виділені регістри та пам'ять. Крім того, ядро додає метадані з інформацією про джерело та призначення, які не можуть бути підроблені. Оскільки відображення віртуальної пам'яті також контролюється ядром, спільна пам'ять між процесами також може підлягати попередньо визначеній політиці. Регулюючи використання примітивів IPC, ядро

арбітражує взаємодію між ізольованими процесами. Він може застосовувати індивідуальну політику контролю для всіх потоків даних на основі специфікації системи.

Користуючись вищезгаданою унікальною особливістю в архітектурі мікроядра, застосовуємо обов'язковий контроль доступу до механізму передачі повідомлень. Обов'язковий контроль доступу – це загальносистемний механізм контролю за дотриманням правил. На відміну від дискреційного контролю доступу, який широко застосовується в комерційних операційних системах, які дозволяють користувачам визначати політику контролю доступу до своїх даних незалежно від глобальних політик, обов'язковий контроль доступу застосовується глобально і не може бути змінений самими користувачами або програмами. Точний обов'язковий контроль доступу реалізований у вигляді модуля ядра, яку навиватимемо матриця контролю даних (МКД) і знаходиться лише в просторі ядра, який недоступний з простору користувача, отже, він захищений від потенційної атаки, якщо ядро заслуговує на довіру.

Ще однією частиною системи, яку потрібно регулювати, є мережевий зв'язок. На відміну від ОС загального призначення, в яких системи зосереджені на забезпеченні справедливого обслуговування всіх процесів, у вбудованих системах змішаної критичності доступ до мережі повинен розглядатися як привілей і надаватися застосунку тільки в тому випадку, якщо це необхідно для досягнення його функціональності. Це пов'язано з тим, що в складній розподіленій системі багато контролерів співпрацюють між собою для виконання завдання управління. Скомпрометований процес може відкривати порти, просто запитуючи сокет, і підробляти або проводити DoS-атаки на інші критичні програми на інших пристроях.

Обмеження мережевих ресурсів підмножини процесів не є тривіальними в ОС загального призначення через монолітну архітектуру ядра. Втім, цього легко досягти за допомогою мікрокернельної архітектури, особливо за допомогою механізму МКД, оскільки в архітектурах мікроядра системні служби та драйвери

пристроїв виконуються як користувацькі процеси, а системні виклики використовують примітиви IPC ядра таким же чином. Таким чином, щоб розширити політики за допомогою управління мережевим зв'язком, така конструкція позбавляє процеси можливості зв'язку з мережевими службами, за замовчуванням використовуючи МКД. Для програм, яким потрібен доступ до мережі, ця конструкція застосовує шаблон проксі-сервера, вводячи легкий спеціальний проксі-процес для програм із вимогами до мережевого зв'язку. Патерн проксі - це відомий патерн проектування програмного забезпечення. По суті, шаблон проектування проксі відокремлює логіку управління застосуванням та інтерфейс у спробі забезпечити сурогат доступу та захистити компоненти керування від надмірної складності. Проксі-процес є локальним делегатом віддаленого процесу на окремому контролері. Уся мережева комунікація з віддаленими процесами спрямовується через виділені проксі-сервери один на один через IPC, що дозволяє політиці IPC диктувати дійсний віддалений зв'язок. Коли процеси керування САБ потребують використання мережі, трафік перенаправляється спеціальним легким проксі-процесом. Під час цього процесу ядро перевіряє запит IPC відповідно до конкретної політики зв'язку з застосунком. На рис. 2.1 зображено схему, в якій кожен контролер застосовує глобальну політику локально, дозволяючи процесу А обмінюватися даними з процесом В через віддалені проксі.

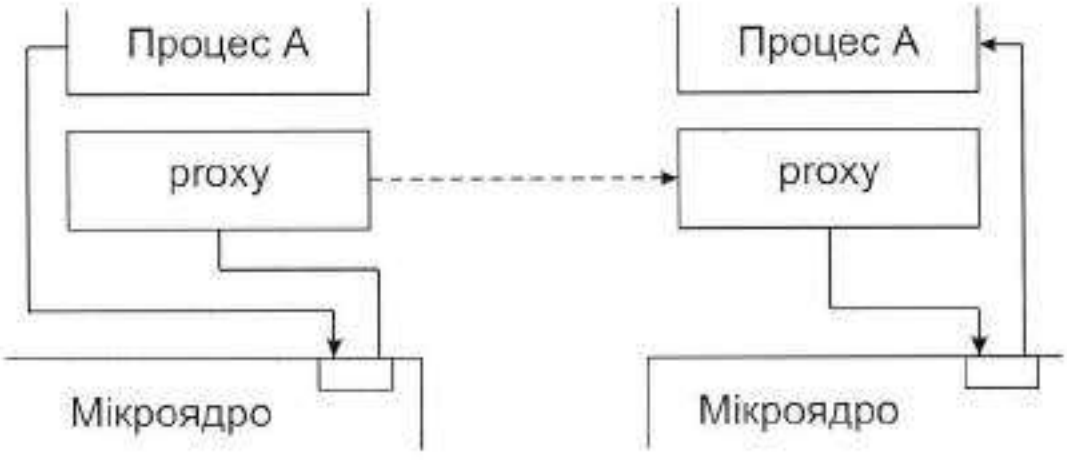


Рисунок 2.1 – Схема обміну даними через віддалені проксі

З іншого боку, мережева безпека автоматизації будівель має два аспекти. Відсутність автентифікації та шифрування в протоколах зв'язку мають великий вплив на безпеку мережевої системи автоматизації будівель. Навіть якщо всі надійні пристрої обмежені доступом до мережі, ніщо не заважає зловмисникам підключати шкідливі пристрої до мережі через фізичний доступ. Ця проблема нетривіальна. Хоча розробка нових протоколів із вбудованою надійною автентифікацією та найсучаснішими алгоритмами шифрування можлива, це навряд чи буде застосовано на практиці через міркування сумісності з існуючими застарілими пристроями та системами. Фактично, остання специфікація мережі вводить керівні принципи підтримки стандартного криптографічного захисту. Однак, навіть нові пристрої не підтримують ці специфікації з міркувань зворотної сумісності. Для того, щоб забезпечити рішення, сумісне з існуючими САБ, але також має можливість захистити ці критичні завдання та пристрої керування від потенційних атак, розглядатимемо для розгортання спарені проксі-сервери. Оскільки кожен мережевий зв'язок має пару проксі-серверів, розгорнутих на обох кінцях, проксі-пара може домовлятися про тунелі безпеки на прикладному рівні незалежно від протоколу зв'язку та автентифікувати один одного. Дані будуть передаватися з процесу САБ до проксі-процесу через IPC. Потім він буде зашифрований і переспрямований до мережевого драйвера. Приймальний контролер перенаправляє вхідний трафік на проксі-сервер для прослуховування, який потім розшифровує та надсилає повідомлення цільовому процесу САБ через IPC.

Дотримуючись філософії шаблону проектування проксі-серверів, а також додавання шифрування отримуватимемо переваги обох цих функцій архітектури: проксі-сервери: тунелювання. Він ізолює логіку управління і регулює використання мережевих ресурсів. Скомпрометовані програми не можуть монополізувати мережу, і вхідні мережеві загрози спочатку будуть перевірені комунікаційною структурою. Він відокремлює функції управління від

комунікаційних. Мережевий зв'язок стає прозорим для застосунків, які потім можна перерозподілити на незалежні пристрої, не турбуючись про взаємозв'язок. Він захищає критично важливі програми від прямого впливу мережі та дозволяє розробникам систем визначати політики прикладного рівня. Замість того, щоб просто пересилати мережеві пакети з/на процеси, в ядро можуть бути вставлені різні специфічні для конкретної області правила для ретельної перевірки вхідної/вихідної інформації.

Таким чином, це дозволяє розробникам систем розглядати безпеку на етапі проектування на високому рівні. Він відображає загальносистемний контроль доступу з локальним контролем доступу на кількох пристроях. Він гарантує наскрізну безпеку навіть у складних середовищах, де розгортаються різноманітні протоколи та застарілі пристрої. Весь зв'язок може бути зашифрований на основі узгодження пари виділених проксі-серверів і тунельований через стандартні протоколи управління.

2.3 Висновки до другого розділу

Здійснено дослідження предметної області та визначено стратегію забезпечення безпеки САБ. В якості основної технології обрана альтернативна архітектура мікроядра з мінімальним функціоналом. Запропонована стратегія відображає загальносистемний контроль доступу з локальним контролем доступу на кількох пристроях. Також, гарантує наскрізну безпеку навіть у складних середовищах, де розгортаються різноманітні протоколи та застарілі пристрої. Весь зв'язок може бути зашифрований на основі узгодження пари виділених проксі-серверів і тунельований через стандартні протоколи управління.

3 МЕТОД СТВОРЕННЯ БЕЗПЕЧНОЇ КІБЕРФІЗИЧНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ ПРИМІЩЕНЬ ПІДПРИЄМСТВ З ВИКОРИСТАННЯМ ОПЕРАЦІЙНИХ СИСТЕМ НА ОСНОВІ МІКРОЯДРА

3.1 Спосіб обробки повідомлень та конфігурування мікроядра

Для прототипу в якості платформ розробки обрано два мікроядра: MINIX 3 і seL4. Завдяки дослідженням архітектура МКД виділяється як основна обчислювальна платформа контролерів будівель. Одним із прикладів є контролер на базі МКД, широко використовуваний контролер автоматизації будівель.

MINIX – це відома ОС на основі мікроядер, яка розроблена як приклад мікроядрового підходу. Останньою версією є MINIX 3, яка націлена на вбудовані пристрої з акцентом на високу надійність. З точки зору користувача, MINIX 3 дуже схожа на традиційну систему в стилі UNIX. Насправді, більшість програм для роботи з користувачем були портовані в MINIX 3. Однак архітектура MINIX 3 повністю відрізняється від традиційних систем, які керуються типовим мікроядром. Ядро MINIX 3 складається всього з 6 000 рядків коду. Код ядра містить апаратні абстракції, переривання, блоки управління процесами, таймери та примітиви IPC. ОС побудована в трьох рівнях: рівень драйверів пристроїв, рівень системних служб; рівень застосунків користувача. Окрім драйверів пристроїв, MINIX 3 складається з кількох серверів, що працюють як ізольовані процеси в просторі користувача, включаючи менеджер процесів, віртуальну файлову систему, менеджер віртуальної пам'яті, службу системної інформації, диспетчер пристроїв тощо. MINIX 3 є найбільш усталеною ОС на основі мікроядра з відкритим вихідним кодом з найбільшою підтримкою драйверів пристроїв. Тому, використовуємо його як одну з платформ для розробки прототипу.

Мікроядро seL4 є найновішим представником сімейства мікроядер L4.

Дотримуючись філософії ядра L4, ядро seL4 підтримує абстракції для віртуальних адресних просторів, потоків і міжпроцесного зв'язку з високою продуктивністю. Найголовніше, що seL4 є першим математично перевіреним програмним ядром. Впроваджена та перевірена формальна верифікація доводить, що виконуваний машинний код, складений із більше, ніж 10 000 рядків коду seL4, є функціонально правильним щодо його високорівневої специфікації за допомогою доведення теорем, що означає, що код ядра вільний від вразливостей, таких як переповнення буфера, невизначена поведінка тощо.

На відміну від MINIX 3 і традиційної Unix-подібної системи, seL4 використовує модель безпеки, засновану на можливостях. З точки зору ядра, ресурси розглядаються як різні типи об'єктів ядра. Права власності або доступу до об'єкта ядра, наприклад, невикористовувані області пам'яті, таблиці сторінок, блоки керування завданнями, кінцеві точки IPC тощо, обліковуються за можливостями. Керування доступом на основі можливостей безпосередньо пов'язане з керуванням віртуальною пам'яттю через MMU. Можливість — це непідробний токен, який представляє явні повноваження власника та безпосередньо керується ядром. Ця модель забезпечує гнучкий механізм для обґрунтування та забезпечення дотримання політики контролю.

У MINIX 3 примітив IPC ядра — це синхронна передача повідомлень. Синхронна передача повідомлень використовує механізм у стилі рандеву. При виклику примітивів IPC процес виклику буде призупинено до тих пір, поки повідомлення не буде скопійовано від відправника до одержувача. Повідомлення є 64-байтовими буферами фіксованого розміру, які включають 4-байтовий ідентифікатор кінцевої точки, 4-байтове поле типу повідомлення та 56-байтове корисне навантаження. Кінцева точка призначення має бути явно вказана для надсилання або отримання повідомлення. Кінцева точка ідентифікує процес унікально серед операційної системи. Він складається з номера слота процесу, об'єднаного з номером генерації для адресації IPC, який зберігається в блоці керування процесом. Є 3 системних виклики: ipc_send() ipc_receive() і

ipc_sendrec(). Системні виклики блокуються до тих пір, поки повідомлення не буде доставлено в процес, що приймає. Системні виклики блокують до отримання повідомлення від цільового процесу. Це забезпечує атомарна операція для надсилання та отримання зв'язку в обидві сторони. У поточній версії, синхронна передача повідомлень зарезервована для драйверів пристроїв і компонентів системного сервера з розробленими протоколами зв'язку.

Примітиви MINIX 3 IPC є ефективним засобом для впровадження обов'язкового контролю доступу для ізоляції процесів та регулювання зв'язку. Модифікуємо ядро MINIX 3 так, щоб передавати примітиви повідомлень усім процесам користувача. Оскільки ядро полегшує всі IPC, воно є ідеальним місцем для забезпечення дотримання політики IPC. Безпосередньо надаючи примітиви IPC всім процесам користувача, також спрощуємо шляхи зв'язку та потік інформації. Крім того, додамо три системні виклики на сервері керування процесами для покращення операцій, пов'язаних з IPC: перетворює ідентифікатор процесу та повертає відповідну кінцеву точку; отримує кінцеву точку процесу за іменем; дозволяє процесу запитувати всі повідомлення, що очікують на розгляд.

Змінимо структуру даних друкованої плати, додавши поле під назвою ідентифікатор контролю доступу і пов'язані системні виклики: fork2(); srv_fork2(). Ці системні виклики можуть призначити кожному процесу, серверу, унікальний номер під час періоду завантаження. Вони призначені для заміни оригінальних системних викликів fork() та srv_fork(() для завантаження серверів процесів та системи із заданими номерами процесів. Ідентифікатори процесів призначаються випадковим чином і можуть змінюватися, тому потрібен цей номер для допомоги у побудові визначень політики IPC. Використовуємо поле, щоб унікально ідентифікувати кожен процес і застосовувати політику контролю.

Здійснимо також реалізацію механізму перевірки політик МКД в примітиві передачі повідомлень. Тепер ядро перевіряє МКД для кожного IPC, щоб визначити, чи дозволено двом процесам обмінюватися даними. Будучи

єдиним кодом, що працює в привілейованому режимі, ядро має абсолютну владу над IPC. Оскільки МКД зберігається в просторі ядра, його не можна легко змінити без перекомпіляції вихідного коду ядра. Таким чином, правильна реалізація МКД у просторі ядра може гарантувати примусову перевірку безпеки.

Як випливає з назви, МКД є табличною структурою даних. МКД використовує розріджену матричну структуру даних для швидкого пошуку та економії простору. Кожна таблиця вказує на запис індексу в матриці. Кожен рядок матриці визначає, з якими процесами може взаємодіяти процес відправки за допомогою передачі повідомлень, і який тип повідомлення дозволений. Тип повідомлення – це число із зазначенням, який тип спілкування дозволений. Інтерпретація типу повідомлення зарезервована для окремих процесів, і ядро передбачає, що воно заздалегідь узгоджене між відправником і одержувачем. У експерименті використовуватимемо поле типу повідомлення для представлення різних віддалених викликів процедур, які певний процес надає іншому процесу для виклику.

Усі повідомлення про підтвердження між процесами були дозволені. Під час виконання ядро буде використовувати МКД для аудиту інформаційного потоку. Оскільки растрове зображення визначено, то повідомлення буде дозволено. З іншого боку, якщо тип повідомлення дорівнює 1, повідомлення буде відхилено, а запит буде відкинуто.

За допомогою МКД архітектуру проксі може бути легко реалізований. За замовчуванням всі процеси заблоковані від мережевих комунікацій. Це досягається шляхом обмеження обміну даними IPC між процесами та мережевими серверами. У прототипі MINIX 3 проксі-серверу призначається той самий номер, що й віддаленому процесу його делегат. Отже, він підпадає під дію однієї і тієї ж політики МКД. Це дозволяє розповсюджувати спільну уніфіковану політику МКД на всі ядра контролерів у мережі. Використовуючи однакову політику МКД для всіх контролерів, процеси можуть бути перенесені на різні контролери, але при цьому забезпечують однаковий рівень регульованої ізоляції

та інтеграції. Таким чином, глобальне забезпечення безпеки може здійснюватися повністю розподіленим способом на кожному локальному контролері. З точки зору системи, інформаційний потік відбувається між процесом А і проксі-процесом, який передається ядром IPC-примітивів, які піддаються МКД. Тому зіставляємо віддалену IPC з локальною IPC, до якої можемо застосовувати політики.

Використовуючи проксі-сервери, можемо ефективно знизити потенційні ризики, викликані шкідливими внутрішніми процесами. Однак іншим набором векторів атак для пристроїв САБ є поширені мережеві атаки, такі як атаки типу "людина посередині", DDoS-атаки, атаки сніфінгу, спуфінг-атаки тощо. Наприклад, якщо зовнішній шкідливий пристрій отримує доступ до мережевого середовища, на цей пристрій не поширюється жодна політика мікроядра, що становить загрозу. Крім того, САБ часто включають кілька протоколів промислового управління одночасно, які часто залежать від постачальника. Складність і залежність САБ унеможливлюють перехід на захищені протоколи на практиці. Більш практичним рішенням було б забезпечити VPN-подібний захищений мережевий тунель між пристроями на САБ за допомогою механізмів аутентифікації та шифрування на прикладному рівні. Це добре вивчена проблема в комп'ютерних мережах. Такі протоколи широко використовуються в Інтернеті та корпоративних мережах. Хоча використання зв'язку на основі проксі-сервера є ідеальним середовищем для створення безпечних мережевих тунелів на основі програми між застосунками. У запропонованому методі використовуватимемо попередньо розподілену пару публічних/закритих ключів у проксі. Симетричний сесійний ключ узгоджується між проксі-парами за допомогою їх асиметричної пари ключів за допомогою стандартного алгоритму обміну ключами.

Це рішення може бути розширене за допомогою спеціальної криптографічної апаратної підтримки, наприклад, за допомогою модуля довірчої платформи для забезпечення міцнішого ланцюга довіри. Наприклад, при

підтримці його авторизована пара кореневих ключів із сертифікатом може зберігати обладнання. Драйвер модуля можна використовувати для надання послуг для проксі-процесів для виведення ключів, запечатування даних та інших криптографічних функцій. У цьому випадку асиметричні пари ключів для проксі-серверів можуть бути отримані під час виконання, і проксі-серверам ніколи не потрібно буде зберігати ключі в енергонезалежній пам'яті.

Додавання всіх цих легких проксі-процесів дозволяє системі зіставляти мережевий зв'язок у локальний IPC, а також дозволяє мікроядру регулювати міжпристрої IPC шляхом арбітражу локальних IPC між локальним процесом і проксі-сервером за допомогою політик МКД. Глобальна безпека досягається завдяки єдиним локальним перевіркам безпеки, які застосовуються на найнижчому рівні в кожному пристрої.

Коли процес надсилає повідомлення проксі-серверу, зв'язок відбувається за такою логікою. Кроки методу:

1) управління САБ обробляє видачу повідомлень для ініціювання зв'язку;

2) запит потрапляє в ядро, а ядро перевіряє валідність цього зв'язку з МКД;

3) ядро пересилає повідомлення проксі-серверу, який очікує на повідомлення;

4) проксі-сервер отримує повідомлення через надіслане йому повідомлення-підтвердження;

5) проксі-сервер перевіряє валідність вмісту повідомлень для конкретної програми;

6) проксі-сервер шифрує і пересилає вихідне повідомлення на мережевий сервер;

7) проксі-сервер відповідає на процес керування САБ, вказуючи на те, що повідомлення було надіслано.

Для передачі великого обсягу даних може використовуватися загальна пам'ять. MINIX 3 підтримує операції спільної пам'яті. Але реалізація реалізується за допомогою IPC-сервера процесу. Щоб налаштувати спільну

пам'ять, процеси обмінюються даними з IPC-сервером за допомогою передавання повідомлень. Внесемо зміни до сервера IPC, щоб він відповідав вимогам політики. Процеси, які ініціалізують спільну пам'ять, явно вказують кінцеву точку процесу, яка може отримати доступ до неї через сегмент пам'яті. Успішне створення сегмента спільної пам'яті повертає випадковий секретний ключ. Процес може передати секретний ключ цільовому вузлу за допомогою стандартного IPC, що може статися лише в тому випадку, якщо їм дозволено обмінюватися даними. Тільки явно вказаний процес з випадковим секретним ключем може приєднати спільну пам'ять. Спосіб, у який спільна пам'ять також опосередковується МКД.

На рис.3.1 зображено пропоновану стратегію з використанням мікроядра.

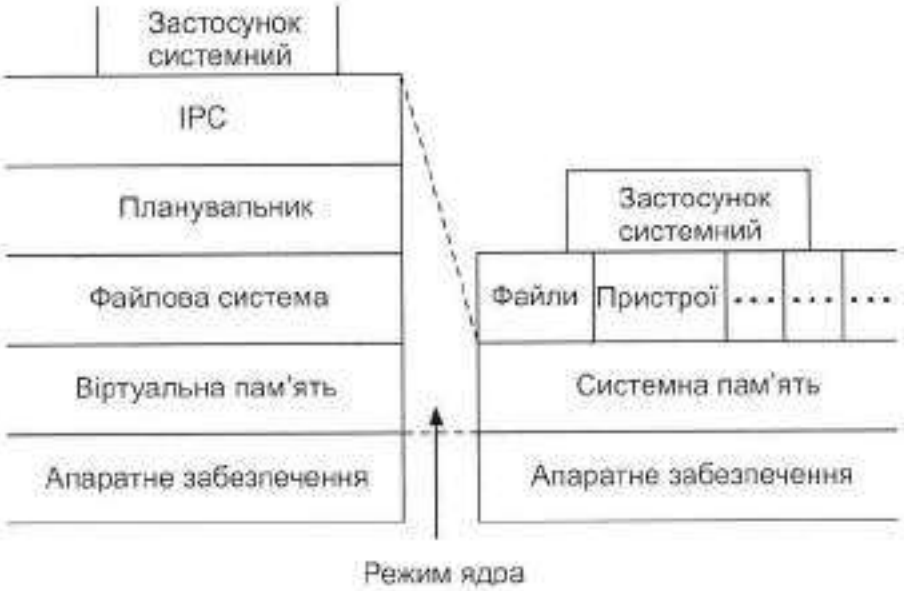


Рисунок 3.1 – Схема режимів ядра ОС

Прототип з використанням мікроядра seL4 має аналогічну реалізацію і досягає тієї ж мети. Завдяки моделі контролю доступу, заснованій на можливостях, мікроядро seL4 вже має надійний контроль доступу. Зміни в просторі ядра не потрібні. Політика реалізується у формі розподілу можливостей кінцевих точок, що відбувається в кореневому ініціальному користувацькому процесі. Мережевий зв'язок на основі проксі-сервера працює так само. Це

свідчить про те, що обрана архітектура добре працює з архітектурою мікроядра в цілому.

Таким чином, розроблено спосіб обробки повідомлень та конфігурування мікроядра. Він реалізується у формі розподілу можливостей кінцевих точок, що відбувається в кореневому користувацькому процесі.

3.2 Метод створення безпечної кіберфізичної системи для автоматизації приміщень підприємств

Розглянемо використання прототипу MINIX 3 з підвищеною безпекою. Оцінимо безпеку системи, ефективність запропонованого рішення, а також продуктивність в порівнянні з немодифікованим мікроядром, а також широко застосовуваним Linux. Поділимо аналіз на три етапи: введення сценарію; модель загроз та аналіз безпеки; аналіз продуктивності.

Для того, щоб реалістично оцінити архітектуру, в якій імітується реалістичний сценарій використання, що відображає суть роботи системи автоматизації будівель введемо два сценарії контролю температури на прикладі системи опалення, вентиляції та кондиціонування повітря в лабораторії для досліджень.

Спочатку розглянемо сценарій контролю температури на одному пристрої. Перший сценарій – це сценарій керування з одним контролером. Метою сценарію є підтримання лабораторної температури в потрібному діапазоні. Якщо не вдасться підтримувати лабораторну температуру протягом тривалого часу, спрацює сигнал тривоги. Адміністратори можуть контролювати стан середовища керування та регулювати бажану температуру лабораторії з веб-порталу.

Для реалізації прототипу в цьому експерименті використовується для імітації контролера будівлі. Датчик температури вибирає барометричний датчик тиску, який може вимірювати як тиск повітря, так і температуру. Привід

обігрівача - це просто привід вентилятора разом з невеликим електричним нагрівальним матом для емуляції. Для приводу сигналізації використовуватимемо бортовий світлодіодний ліхтар. Контейнер включає в себе дві перегородки. Один з них використовується для моделювання лабораторії. Інше використовується для імітації приміщення, де монтується вентилятор.

Завдання управління реалізується в п'яти процесах: драйвер датчика температури, драйвер приводу вентилятора, драйвер приводу сигналізації, процес контролю температури та процес веб-інтерфейсу відповідно. Кожен процес обмінюється даними за допомогою синхронної передачі повідомлень. Крім того, кожен процес має набір попередньо визначених типів повідомлень, які він приймає, тоді як механізм МКД обмежує, які типи повідомлень можуть надсилатися з якого процесу. Процес датчику температури періодично відбирає зразки температури навколишнього середовища та надсилає свіжі дані за допомогою неблокуючого виклику системи надсилання до процесу контролю температури. Як процес приводу сигналізації, так і процес приводу вентилятора налаштовані на пасивне очікування команд від процесу контролю температури. Нарешті, процес веб-інтерфейсу діє як базовий людино-машинний інтерфейс. Це статичний веб-сервер HTTP з 5 фіксованими дочірніми потоками. Процес підтримує сокет TCP на порту 8080. У сценарії найважливішим процесом є процес контролю температури. Коли процес запускається, то він виконує функцію ініціалізації, щоб отримати кінцеву точку кожного процесу, з яким йому потрібно зв'язатися. Потім процес переходить у цикл while, очікуючи нових даних датчика від процесу датчика температури. Коли надходять дані, дані датчика порівнюються із заданим значенням температури, щоб вирішити, вмикати чи вимикати вентилятор, тим часом буде перевірено таймер, якщо температура виходить за межі заданого значення. Якщо протягом певного часу регулятор температури не зможе підтримувати діапазон у межах заданого значення, спрацює сигнал тривоги. Потім процес перевірить, чи є очікуючі повідомлення від процесу веб-інтерфейсу для оновлення нового заданого

значення. Наприкінці циклу while інформація про середовище буде записана у файл журналу.

Розглянемо мережевий сценарій контролю температури. Другий експеримент – це сценарій контролю температури з декількома пристроями. Аналогічно попередньому прикладу, метою цього контрольного завдання є регулювання температури в лабораторії. Прагнучи зробити симуляцію реалістично рефлексивною САБ, і перевірити мережеву безпеку платформи, ця схема контролю температури реалізована в контролерах. Контролер А представляє контролер будівлі та містить процес веб-інтерфейсу, процес контролю температури та драйвер приводу сигналізації. Контролер В представляє блок обробки повітря та містить драйвер датчика температури та драйвер приводу вентилятора.

Між тими процесами, які обмінюються даними через мережу, проксі-сервери сполучаються з попередньо розподіленими парами відкритих/закритих ключів. Ці проксі-сервери включають проксі-сервер контролю температури в контролері В.

Реалізація на Linux дуже схожа на реалізацію на MINIX 3. Єдиною суттєвою відмінністю є те, що у Linux міжпроцесний зв'язок здійснюється за допомогою черг повідомлень POSIX. Черги повідомлень забезпечують асинхронний протокол зв'язку. Більшість операційних систем реального часу заохочують використання черги повідомлень як основного механізму зв'язку між процесами для застосунків у реальному часі. У Linux черги повідомлень є першими. Вони реалізуються через віртуальну файлову систему і підтримуються бібліотекою реального часу. Крім того, відсутні проксі-процеси для мережевого зв'язку. Кожен процес може безпосередньо відкривати мережевий сокет.

Далі проаналізуємо моделі загроз та здійснимо аналіз безпеки. Щоб повністю перевірити життєздатність запропонованої системи зв'язку застосунків, змоделюємо можливі вектори атак і порівняємо результати з реалізацією в традиційному стилі. Грунтуючись на існуючих реальних атаках,

модель загроз передбачає, що зловмисники можуть мати доступ до мережі САБ, а деякі процеси можуть містити вразливості, якими можна скористатися. Відповідно до моделі загроз, змодельовано три категорії атак від найменш початкового до найбільш грамотного зловмисника.

Типи атак: мережеві атаки; атаки на скомпрометовані мережеві інтерфейси; атаки на скомпрометовані процеси контролю. У кожному з цих сценаріїв аналізуємо, які привілеї можуть отримати зловмисники, щоб перевірити властивості безпеки/захисту конструкції зв'язку.

Аналіз припускає, що процес приводу сигналізації, процес контролю температури є критично важливими застосунками для системи. Крім того, припускаємо, що проксі є надійними компонентами, які є функціонально правильними. На практиці вважатимемо, що це безпечне припущення через просту функціональність проксі-серверів і потенціал для автоматичної генерації відповідно до високорівневої специфікації.

Мережеві атаки створюють великі проблеми для розподілених САБ. Багато атак спеціально націлені на слабкі місця в промислових протоколах управління та зв'язку. Проаналізуємо три типи атак: атаки з підміною особистості; атаки типу "людина посередині"; DoS-атаки. Ці атаки становлять більшість загроз для САБ. Враховуючи сценарій з двома пристроями, встановимо третій пристрій у мережі за допомогою Kali Linux для виконання мережевих атак. Багато промислових протоколів зв'язку можуть підтримувати маршрутизацію трафіку через TCP/IP та Ethernet. У традиційній реалізації зловмисник в одній і тій же мережі може довільно відправляти спуфінг-пакети на керуючі пристрої. Без розгортання проксі-сервера помилкові дані про перепад тиску, надіслані в процес керування повітряним потоком, втручатимуться в логіку керування.

Однак цій атаці вдалося запобігти за допомогою прототипу. Це пов'язано з тим, що проксі-сервер обробляє автентифікацію та шифрування між пристроями. Ті пакети підміни, які не були зашифровані правильним ключем

сеансу, використовуються між проксі-сервером контролю температури та проксі-сервером датчика температури, отже, повідомлення про підміну не може бути автентифікованим, і його просто було видалено або повідомлено проксі-сервером. Аналогічно, при правильній реалізації захищеного мережевого тунелю, за умови, що криптографічна інфраструктура реалізована правильно, зловмисники не зможуть проводити атаки типу "людина посередині" без отримання сертифікованої асиметричної пари ключів. Максимум, що може зробити зловмисник, це просто переслати повідомлення. Для тих широкомовних проксі-серверів, які обробляють застарілі процеси, залежно від конкретної реалізації, повідомлення може бути не зашифровано. Крім того, проксі-сервер може включати підхід до білого списку, щоб запобігти отриманню спуфінгових широкомовних повідомлень.

Якщо контролер підтримує модуль довірчої платформи, то проксі-сервер може скористатися непідробним ідентифікатором пристрою та використовувати його для перевірки розбірливості відправників. Ключовим моментом є те, що архітектуру проксі-сервера забезпечує рівень абстрактності для автентифікації та дотримання політик на системному рівні, і може бути адаптований за потреби за допомогою нових технологій. Таким чином, цілісність логіки управління та конфіденційність даних/команд пристрою зберігаються із запропонованим прототипом.

З іншого боку, однією з найвпливовіших атак на САБ є DoS-атаки. DoS-атака спрямована на виснаження обчислювальних і мережевих ресурсів у системі. Отримавши доступ до мережі, зловмисники можуть наповнити мережу та тимчасово порушити зв'язок між контролером лабораторії та контролером камери. Хоча зловмисники, можливо, не зможуть проникнути в систему керування, за допомогою DoS вони можуть запобігти або затримати дані датчиків, які будуть доставлені процесом контролю температури через мережу. Без проксі-сервера процес управління повітряним потоком був би зайнятий обробкою всіх цих помилкових запитів з мережі. Однак, використовуючи

запропоновану структуру зв'язку, це безпосередньо вплине лише на проксі-сервери, контур керування в процесі керування повітряним потоком залишиться недоторканим. Незважаючи на те, що він не може отримувати поточну лабораторну температуру в режимі реального часу, він все одно зберігає контроль і може вмикати сигналізацію або регулювати вентилятор. Звичайно, якщо процес приводу сигналізації також є віддаленим процесом, це також може вплинути. Ця гарантія залежить від того, як система інтегрована та розгорнута. Але в цілому проксі працюють як брандмауери для конкретних застосунків і фільтрують недійсні запити з високою точністю. Крім того, хоча для повного виключення наслідків DoS-атаки необхідний додатковий захист на мережевому рівні, то проксі-сервер може допомогти виявити несанкціонований мережевий трафік відповідно до системної політики та повідомити про помилкову поведінку з боку САБ.

Як і більшість мережевих застосунків, кібератака є постійною загрозою. Якщо зловмисник може знайти неправильну конфігурацію або вразливості в процесі веб-інтерфейсу, то зловмисник може довільно контролювати процес. З цим припущенням змоделюємо поведінку скомпрометованого процесу веб-інтерфейсу. По-перше, зловмисник може спробувати надіслати повідомлення про підробку одноранговим процесам, наприклад, процесу приводу сигналізації та процесу приводу вентилятора. Із запропонованою структурою зв'язку та базовим обов'язковим контролем доступу, спроба спуфінгу, націлена на будь-які процеси, крім явно авторизованого процесу з явно авторизованим типом зв'язку, він просто зазнає невдачі. Таким чином, цілісність висококритичних застосунків зберігається навіть за наявності скомпрометованих процесів.

Крім того, якщо припустити, що процес контролю температури є програмою з високою гарантією, отже, перешкоди, спричинені скомпрометованим веб-інтерфейсом, дуже обмежені. Однак у традиційній системі зловмисник може викликати системні виклики, такі як kill, fork тощо, але через ізоляцію, передбачену політикою МКД, обмеження на системні

виклики накладатимуться навіть на веб-інтерфейс із привілеями root, що, є лише концепцією у файловій системі для архітектури мікроядра. Хоча для того, щоб повністю мінімізувати вплив такого рівня компрометації, застаріла програма повинна працювати в обмеженому ресурсами розділі з планувальником реального часу. Крім того, проксі-сервер може бути розроблений з механізмом моніторингу, у цьому випадку, навіть якщо інтерфейс користувача продовжує підробляти проксі зайвими повідомленнями на високій швидкості, недійсний зв'язок може бути просто припинений, а порушення можна зареєструвати та повідомити про порушення. У кращому випадку проксі буде завантажуватись в помилкових повідомленнях, але основна логіка управління залишиться незмінною.

Процеси на низькому рівні, такі як процес датчика температури та процес приводу сигналізації, також мають шанси бути скомпрометованими. Такі порушення будуть руйнівними в будь-якій ситуації. Те ж саме і з пропонованим прототипом. Ця проблема потребує додаткових рішень, наприклад, шляхом розгортання резервних датчиків і виконавчих механізмів. Однак на практиці логіка контролю негативного лабораторного тиску, швидше за все, поєднується з іншою логікою управління на тому ж пристрої, наприклад, пожежною сигналізацією. Навіть у ситуації, коли процес приводу сигналізації скомпрометовано, прототип фреймворку зв'язку застосунків запобігає подальшій ескалації зловмисником своїх привілеїв на пристрої та в мережі.

Постановка експерименту показує, як потенційні ризики поширених мережевих атак можна запобігти або мінімізувати без негайної заміни існуючих протоколів промислового керування та застарілих програм. Навіть якщо зловмисникам вдасться скомпрометувати деякі процеси, вони не матимуть повної влади. Отже, жоден пристрій не відповідає виключно за безпеку всієї системи.

У САБ час відгуку та затримка мають велике значення в реальному використанні. Таким чином, будь-які нові функції або конструкції повинні бути

зважені з точки зору продуктивності з порівнянням існуючих систем. Безпека не дається безкоштовно. Архітектуру додає ряд проксі-процесів і криптографічне тунелювання, а також кілька перемикань контексту під час операцій. Отже, розумно бачити помірно збільшені накладні витрати в порівнянні з системою без використання шифрування, якщо продуктивність прийнятна в середовищі управління САБ.

Історично склалося так, що однією з головних проблем при прийнятті архітектури мікроядра була її нижча продуктивність у порівнянні з монолітними архітектурами. Однак це проблемне питання було мінімазовано сучасним апаратним забезпеченням і вдосконаленням архітектури ОС. Сучасні мікроядра, такі як seL4, досягають високої продуктивності за рахунок безлічі методів оптимізації коду. Отже, ці специфічні недоліки можуть не застосовуватися однаково до систем мікроядра в домені САБ. Зокрема, було продемонстровано, що характерний еталон систем мікроядер, затримка IPC, становить менше 100 наносекунд для деяких реалізацій. Загалом, продуктивність IPC є найважливішим показником продуктивності мікроядра, оскільки всі взаємодії компонентів відбуваються через IPC. Оскільки мікроядра сильно налаштовані на продуктивність IPC, цей аналіз висунув гіпотезу, що додаткові витрати на проксі-процеси будуть в межах допустимих для контролерів автоматизації будівель.

На основі спостережень за трафіком САБ з локальних університетських мереж САБ-пристрої обмінюються даними в мережі за допомогою протоколів. На прикладному рівні блок даних протокол інкапсульований у частині даних. Максимальна довжина може досягати великого значення, однак для зворотної сумісності розмір пакета зазвичай обмежений 480 байтами, що вказується в кожному полі.

Таким чином, ця оцінка оцінює лише продуктивність затримки мережі до 1024 байт у кожній ситуації. Крім того, трафік САБ відносно розріджений з близько 35 пакетами/секунду для магістралі і включає близько 40 пристроїв.

Через ці спостереження з низькою пропускною здатністю, оцінка не вимірювала пропускну здатність системно, хоча могла перевищити 35 пакетів на секунду.

3.3 Висновки до третього розділу

Розроблено спосіб обробки повідомлень та конфігурування мікроядра. Його реалізація базується на формі розподілу можливостей кінцевих точок, що відбувається в кореневому ініціальному користувацькому процесі.

Розроблено метод створення безпечної кіберфізичної системи для автоматизації приміщень підприємств. Він базується на архітектурі мікроядра.

4 ОРГАНІЗАЦІЯ ЗВОРОТНЬОЇ СУМІСНОСТІ І ВІРТУАЛІЗАЦІЇ

4.1 Реалізація зворотньої сумісності та віртуалізації

Тенденція до більш інтелектуального управління в САБ вимагає інтеграції даних і нових можливостей в пристроях управління. Для того, щоб задовольнити цю вимогу, сучасний САБ вимагає більшої обчислювальної потужності. З розвитком апаратного забезпечення багато постачальників контролерів автоматизації поступово переходять на архітектуру як основну обчислювальну платформу поточних і майбутніх контролерів будівель. У відповідь на ці розробки апаратні технології САБ кардинально змінилися від 8-розрядних до 64-розрядних процесорів, від одноядерних до багатоядерних. Поряд з логікою управління програмним забезпеченням, хмарні клієнти, веб-сервер, аналітика часто розгортаються на одних і тих же вбудованих контролерах, що підвищує складність системи і робить її схильною до потенційних вразливостей і експлойтів.

Незважаючи на те, що в дослідженні безпеки є багато перспективних пропозицій, дуже рідко можна побачити, як ці рішення використовуються в реальних продуктах. Завдання модернізації існуючої САБ для задоволення вимог змішаної системи критичності є двояким. По-перше, безпосереднє застосування безпеки в застарілій системі є неможливим через апаратні обмеження та відсутність міркувань безпеки при початковому проектуванні. За останні 10 років було розгорнуто багато існуючих контролерів будівель, наприклад, програмовані логічні контролери. Ці контролери часто не розрізняють режим користувача та режим привілеїв. Крім того, в цих старих моделях різні завдання управління вбудовані в один і той же простір пам'яті без операційної системи, щоб допомогти ізолювати один одного. З огляду на їх вік, модернізувати такі пристрої по суті неможливо. Викриття цих пристроїв в Інтернеті пов'язане з великими ризиками.

Повна заміна застарілої системи є непомірно дорогою. Перенесення

існуючого програмного забезпечення є дорогим і трудомістким процесом. Дослідження показує, що в бізнесі перенесення існуючого програмного забезпечення на інші платформи, як правило, коштує 30% від загальної вартості розробки продукту. Багато постачальників мають власні апаратні платформи з індивідуальними операційними системами для підтримки свого програмного забезпечення. Наприклад, для відомого рішення автоматизації використовують вбудований контролер IoT, для якого розроблено своє програмне забезпечення на відносно новому процесорі ARM і операційну систему реального часу, яка використовується протягом останнього десятиліття. Це рішення, ймовірно, є одним з найбільш схожих комерційних застосувань у порівнянні із запропонованим нами прототипом ОС з підвищеною безпекою з точки зору апаратної платформи та архітектури ОС. Незважаючи на схожість, нетривіально перенести існуючу кодову базу на запропоновану ОС з підвищеною безпекою, не кажучи вже про перенесення багатьох старих програм на інші рішення. З іншого боку, новим ОС часто не вистачає підтримки драйверів пристроїв від сторонніх постачальників, що ще більше ускладнює процес прийняття.

Ці виклики підтверджуються практикою. Розробники САБ підтримують споживчі та промислові рішення для автоматизації будівель IoT. Прагнучи інтегрувати продукти управління з хмарними платформами для забезпечення «розумних» підключених активів і розширеної аналітики, ядром цього рішення є розумний шлюз IoT. Шлюз відповідає за збір даних пристрою та зв'язок із хмарними платформами. Отже, безпека шлюзу має вирішальне значення, оскільки компрометація шлюзу дозволить атакам отримати доступ до хмарних активів і керувати виконавчими пристроями.

Тому, краще побудувати такий шлюз за допомогою вищезгаданої платформи мікроядра безпеки. В осяжному майбутньому нереально відмовитися від існуючої роботи на користь кращої безпеки. Отже, будь-яке практичне рішення має бути ненав'язливим і сприяти цьому переходу в поступовому поступовому процесі. Відмовлятися від застарілих стеків програмного

забезпечення для бізнесу непомірно дорого. Неможливо очікувати, що компанії створюватимуть застосунки з нуля на користь безпеки.

Одним із рішень для конвергенції застарілого програмного забезпечення та нової платформи є віртуалізація. Віртуалізація відноситься до технологій, призначених для забезпечення віртуального середовища виконання для розміщення програмного забезпечення, яке традиційно працює на фізичних комп'ютерах. Віртуалізація була ключовим фактором технології хмарних обчислень. Це дозволяє різним операційним системам ефективно використовувати однакові апаратні ресурси та зберігати незалежність. Цей метод розділяє фізичні ресурси на кілька часток, що робить застосунки більш доступними, зменшує кількість необхідних ресурсів, забезпечує краще управління і, таким чином, скорочує витрати. Найголовніше, що програми не повинні залежати від конкретної ОС.

З міркувань вартості та безпеки переважає віртуалізація на вбудованій системі на кристалі. Тим більше, що з передовими апаратними асистентами вплив на зниження продуктивності є мінімальним. Основна ідея віртуалізації полягає в тому, щоб абстрагувати апаратне забезпечення, наприклад, процесор, пам'ять, мережу і т.д., в ізольовані середовища виконання, щоб дозволити кожному окремому середовищу працювати так, ніби на приватному комп'ютері. Віртуалізація приносить бажану користь вбудованим пристроям. Віртуалізація дозволяє консолідувати кілька різнорідних хостів з багатими функціональними можливостями на одному процесорі. Використання вбудованої віртуалізації не тільки знижує витрати, але й забезпечує можливість розміщення застарілих програм. Найсильнішою мотивацією для вбудованої віртуалізації є те, що вона підвищує безпеку САБ шляхом ізоляції програмного забезпечення в середовищах виділення. Ця ідея спеціально підходить для систем змішаної критичності в САБ. Якщо ізоляція між віртуальними середовищами може бути гарантована, а політика зв'язку може суворо застосовуватися відповідно до специфікації, то критичні процеси в САБ матимуть власний виділений

віртуальний процесор і власний виділений віртуальний простір пам'яті, навіть якщо вони розміщені спільно в одних і тих же фізичних контролерах. Вплив потенційних порушень можна мінімізувати, запустивши ці контрольні завдання у віртуальній машині. Ще краще те, що застаріле програмне забезпечення може вимагати дуже мінімальних модифікацій.

У загальному випадку вбудована віртуалізація на архітектурі ARM може бути забезпечена двома способами: пара-віртуалізація; апаратна віртуалізація. Пара-віртуалізація — це технологія віртуалізації, яка використовує модифіковану гостьову операційну систему у співпраці з гіпервізором для імітації середовища виконання. При паравіртуалізації гостьова ОС працює в користувачів як звичайний процес. Гіпервізор працює в просторі ядра. Гостьова ОС зв'язується з гіпервізором через абстрактний шар, який називається монітором віртуальної машини (VMM). VMM надає гостьовій ОС додаткові API, які називаються гіпервикликами для доступу до послуг, пов'язаних з апаратним забезпеченням, таких як керування віртуальною пам'яттю, переривання та мережі. Оскільки очікується, що звичайні ОС працюватимуть у просторі ядра, який безпосередньо керує апаратним забезпеченням, і ці ОС повинні бути явно перенесені для використання API VMM. Перевага цієї методики полягає в тому, що не потрібне спеціальне обладнання, і вона відносно простіша та ефективніша порівняно з повністю змодельованим обладнанням нижчого рівня, оскільки ОС та гіпервізор, які працюють разом, щоб зберегти роботу для емуляції ресурсів системи. Такі проекти, як Xen, L4Re і VMware підтримують цей підхід. Однак модифікація звичайних ОС не є тривіальною. Крім того, оскільки гостьова ОС не може отримати прямий доступ до апаратних пристроїв, драйвери в гостьовій ОС повинні бути відповідним чином змінені. VMM і управління пам'яттю в гіпервізорі додають додаткові витрати, що збільшує вплив на продуктивність, в порівнянні з роботою рідної системи безпосередньо на апаратному забезпеченні. Паравіртуалізація має недолік у вигляді високих витрат на інженерні розробки, пов'язані з необхідністю модифікації гостьових ОС [72]. Інший підхід –

використання апаратної віртуалізації. Апаратна віртуалізація покладається на додаткову апаратну підтримку для створення віртуальних середовищ. Перевага цього підходу полягає в тому, що він дозволяє використовувати немодифіковані звичайні ОС у віртуальних машинах з мінімальними додатковими витратами. Вимагаючи вимог віртуалізації, як ARM, так і архітектура X86 забезпечили платформу, яка підтримує віртуалізацію за допомогою апаратного забезпечення. З виходом ARM тепер підтримується розширення віртуалізації у своїй архітектурі. Ці розширення складаються з трьох основних частин: розширення віртуалізації процесора; віртуалізація пам'яті; розширення віртуалізації введення-виведення. Віртуалізація процесора додає новий привілейований режим процесора, режим рівня привілеїв. Цей режим має вищі привілеї, ніж звичайний режим ядра, що дозволяє гіпервізору виконувати конфіденційні інструкції. Крім того, це розширення додає додатковий набір векторних таблиць і регістрів для імітації виділених віртуальних процесорів, на яких гостьові ОС можуть працювати, не відпускаючи під час перемикання контекстного режиму. Завдяки підтримці віртуалізації пам'яті, ARM забезпечує кілька рівнів трансляції адрес пам'яті, безпосередньо реалізованих в апаратному забезпеченні. Така реалізація спрощує роботу гіпервізора з управління віртуальною пам'яттю та підвищує продуктивність. Використання програмного забезпечення для емуляції пристроїв вводу/виводу є дорогим, розширення вводу/виводу забезпечує віртуальну обробку переривань для полегшення цього процесу. Використовуючи віртуальний розподільник переривань, гіпервізор може передавати конкретні обробки переривань безпосередньо на віртуальні машини без додаткових витрат на маршрутизацію та відстеження кожної операції введення-виведення. Для деяких останніх моделей ARM це навіть додано системний пристрій, який дозволяє гіпервізору безпечно налаштовувати прямий доступ до пам'яті (DMA) з гостьовими ОС, що зменшує ризик того, що шкідливі драйвери DMA заповнять фізичну пам'ять. Цей підхід є кращим для вбудованої віртуалізації не тільки тому, що він знижує складність реалізації гіпервізора і

забезпечує віртуальним машинам майже мінімальні накладні витрати на продуктивність. Найголовніше, що такий підхід підвищує надійність і безпеку системи.

Рішення гіпервізора показує перспективний напрямок для підтримки потреби в захищеному контролері САБ на практиці. Однак безпека контролерів САБ не пов'язана з віртуалізацією. При такому підході вирішальне значення має функціональна коректність і надійність гіпервізора. Оскільки система залежить від гіпервізора, щоб гарантувати ізоляцію, будь-які несправності в гіпервізорі загрожуватимуть функціональності критичних завдань керування. Так як гіпервізор працює в режимі, то при наявності вразливостей в гіпервізорі їх можна експлуатувати, що дасть зловмисникам більш високі привілеї, ніж саме ядро.

Одним з популярних гіпервізорів є фреймворк віртуалізації, який перетворює ядро Linux на гіпервізор. Він має перевагу у вигляді широких сервісів та підтримки драйверів. Але через монолітну конструкцію ядра зазнає тих же загроз, що і Linux. Нещодавні дослідження показали, що можуть бути використані багато вразливостей, таких як екранування виконання гостьової системи, читання гостьовими іншими гостьовими даними, підвищення привілеїв від кільця 3 до кільця 0, seL4 як мікровізор.

Враховуючи, що монолітні ядра за своєю суттю неможливо захистити без шкоди для зручності використання та сумісності, підхід мікроядра представляє альтернативне рішення. Для того, щоб зробити припущення про безпеку обґрунтованим, базовий гіпервізор повинен бути надійним, функціонально правильним без невизначеної поведінки і, бажано, мати якомога менший розмір. З цих причин формально верифіковане мікроядро seL4 є відмінним рішенням для реалізації гіпервізора на основі мікроядра, так званого мікровізора.

Мікроядро — це мінімальна системна база для забезпечення апаратної абстракції для побудови різних системних служб. Гіпервізор призначений виключно для управління базовим апаратним забезпеченням для

мультиплексування ресурсів між різними віртуальними середовищами виконання, яке включає як служби користувацького призначення, так і код ядра. Завдяки схожості функціоналу та структури, мікроядро може бути використане для реалізації гіпервізора. Сімейство мікроядер L4 було побудовано як мікровізор для підтримки як пара-віртуалізації, так і апаратної віртуалізації протягом тривалого часу. Їх реалізації продемонстрували належні результати як гіпервізори в промисловості та наукових колах. Однак жодне з існуючих мікроядер не може забезпечити такий же рівень гарантії в порівнянні з seL4. Розроблено мікровізор на базі seL4 для процесора ARM. Використовуючи апаратну віртуалізацію, випустили бібліотеку для побудови VMM. Тому, мікровізор seL4 використовується як фундаментальна платформа для реалізації прототипу контролера САБ.

Мікровізор ізолює завдання управління в різні віртуальні машини, але в домені САБ багато завдань управління також повинні взаємодіяти та співпрацювати разом для виконання повсякденних операцій. Тому, необхідний обов'язковий механізм контролю доступу, який дозволяє розробникам застосовувати тонкі політики комунікації лише щодо тих завдань, які дозволені до співпраці. Модель доступу, заснована на можливостях seL4, забезпечує гнучкий метод прозорого посередництва загальносистемних політик безпеки між віртуальними середовищами. У архітектуруі seL4, заснованому на можливостях, все є об'єктом. Кожен об'єкт ядра представляє частку ресурсів, наприклад, сторінки пам'яті, прив'язані до пристрою області, канали зв'язку тощо. Можливість є непідроблюваним токеном із пов'язаними правами доступу, який посилається на відповідний об'єкт ядра. Кожна можливість може бути призначена для конкретного процесу. Ядро керує токеном можливостей від імені процесів. Коли створюється об'єкт, ядро повертає індекс у таблиці можливостей процесу власника, який може бути використаний процесом для посилання на фактичні об'єкти ядра та виклику його функцій. Ядро захищає цю таблицю можливостей, тому її неможливо підробити.

Для того, щоб два процеси могли взаємодіяти один з одним, процеси повинні володіти можливостями каналу зв'язку, кінцевої точки IPC. Отже, якщо процеси не володіють такою здатністю, Вони не можуть спілкуватися. По суті, загальносистемні політики безпеки застосовуються у формі розподілу можливостей, які можуть бути попередньо налаштовані статично. Щоб полегшити розробку, seL4 надала предметно-специфічну мову для опису розподілу можливостей систем seL4. Вона зчитує специфікацію і переводить її в низькорівневий код C. Вона використовується для створення та розповсюдження як можливостей кінцевих точок IPC, так і спільних сторінок пам'яті між процесами та віртуальними машинами. Згенерований дистрибутив можливостей вбудовано в перший процес, який також служить завантажувачем процесів під час завантаження.

Використовуючи мікровізор seL4 та обов'язковий контроль доступу, що примусово контролюється політикою, можемо налаштувати два середовища виконання, рідний розділ seL4 та віртуальну машину Linux. Рідний розділ - це середовище виконання, яке безпосередньо працює поверх seL4 з доступом до API ядра. Розділ забезпечує надійну ізоляцію та вищу безпеку завдяки невеликому TCB, а також моделі доступу на основі можливостей високої впевненості, що забезпечується безпосередньо ядром seL4. Застосунки в цьому розділі залежать тільки від тонкого шару коду у вигляді процесу seL4 з підтримкою існуючих системних служб і драйверів пристроїв. Втім, з тієї ж причини, що й вище, поки що не існує сумісних із POSIX системних викликів, а драйвери пристроїв обмежені. Отже, застосунок має бути реалізований унікальним способом. З іншого боку, розділ Linux забезпечує багате середовище виконання для існуючих застарілих або низькокритичних програм. За допомогою апаратного забезпечення немодифіковане ядро Linux працює як процес seL4. Оскільки звичайне ядро Linux не зв'язується з бібліотеками seL4 і виконується у власному просторі пам'яті, воно не знає про базовий seL4, і тому неможливо взаємодіяти з програмами, запущеними в рідному розділі.

Застосунки в середовищі Linux такі ж, як якщо б вони працювали в звичайній системі Linux. Вони можуть отримати доступ до стандартних API, системних викликів, бібліотек тощо, і більшість із них можуть виконуватися без перенесення. Крім того, деякі драйвери пристроїв у ядрі Linux також можуть бути повторно використані, щоб полегшити відсутність драйверів пристроїв у seL4.

Примусовий зв'язок між розділом Linux і рідним розділом може бути досягнутий за допомогою завантажуваного модуля ядра. Модулі ядра Linux — це фрагменти двійкового коду, які можна динамічно завантажувати та вивантажувати під час виконання. Модулі ядра Linux запускаються в ядрі Linux простір. Це розширює функціональність ядра. У цій конфігурації можна побудувати модуль ядра зв'язку, який знає модель можливостей seL4 та виклики ядра. Коли застосунку в Linux потрібно встановити зв'язок із рідними службами або процесами, програма може викликати зареєстровані системні виклики, і за допомогою цього модуля ядра можуть надсилатися або отримувати повідомлення або дані з кінцевої точки IPC.

Таким чином, це рішення збалансовує зручність використання та безпеку, розміщуючи як середовища виконання Linux, так і природнє середовище високої надійності для практичних цілей. В спробі полегшити процес міграції з традиційного Linux загального призначення на керовану мікроядром систему високої надійності для контролерів в домені САБ, це рішення дозволяє інкрементну стратегію спочатку на конкретних функціях, дозволяючи при цьому продовжувати створювати решту системи.

4.2 Реалізація прототипу

Розглянемо реалізацію запропонованого прототипу з використанням мікровізора seL4. Використовуючи мікровізор seL4 з відкритим вихідним кодом, це рішення налаштовує платформу з двома середовищами виконання, одним

розділом Linux, реалізованим налаштованою віртуальною машиною Linux, і одним рідним розділом з природніми процесами seL4. Крім того, впроваджено безпечну систему завантажувального ланцюга для захисту цілісності програмного забезпечення. Процедура безпечного завантаження застосовує криптографічні методи в процесі завантаження, які перевіряють цифровий підпис кожного компонента та гарантують, що лише авторизовані компоненти можуть бути завантажені на платформу.

Використовуючи рішення мікровізора на основі концепції seL4, прототип структуровано. Наявні такі основні компоненти: завантажувач; монітор віртуальної машини; гостьова ОС Linux; сервер зв'язку. Завантажувач є кореневим процесом та ініціалізатором користувача для системи seL4. Частина інструментарію містить функції, які перекладаються безпосередньо з мови під час компіляції. Він бере опис зі згенерованих файлів С зі специфікації системи та викликає ядро seL4 для розподілу ресурсів, таких як створення кінцевих точок IPC, блоків керування процесами та розподілу можливостей потрібному власнику. Ще однією метою цього процесу ініціалізації є завантаження програм у систему в об'єднаному архіві. Нарешті, після того, як всі завдання під час процесу завантаження виконані, цей модуль запускає всі інші процеси і переходить у сплячий режим.

Розроблене безпечне середовище виконання з використанням мікровізора seL4 зображене на рис. 4.1.

Ще одним важливим компонентом є монітор віртуальної машини (VMM). VMM – це програмне забезпечення, яке створює віртуальне середовище. У прототипі ВММ реалізована з використанням процесу з seL4. Його розроблено з метою полегшити процес створення застосунків і серверів за допомогою мікроядра seL4. Завдяки унікальності seL4 вона відрізняється від традиційної Unix-подібної системи. Зокрема, застосунки в системі на основі seL4 вимагають частого використання IPC, вручну реалізуючи всі взаємодії процесів і відстежуючи можливості, що не є тривіальним. Він дотримується

компонентного підходу, який моделює систему як набір взаємодіючого програмного забезпечення та надає різні попередньо визначені модулі для розробників, щоб абстрактно вказати зв'язок, не турбуючись про детальну реалізацію.

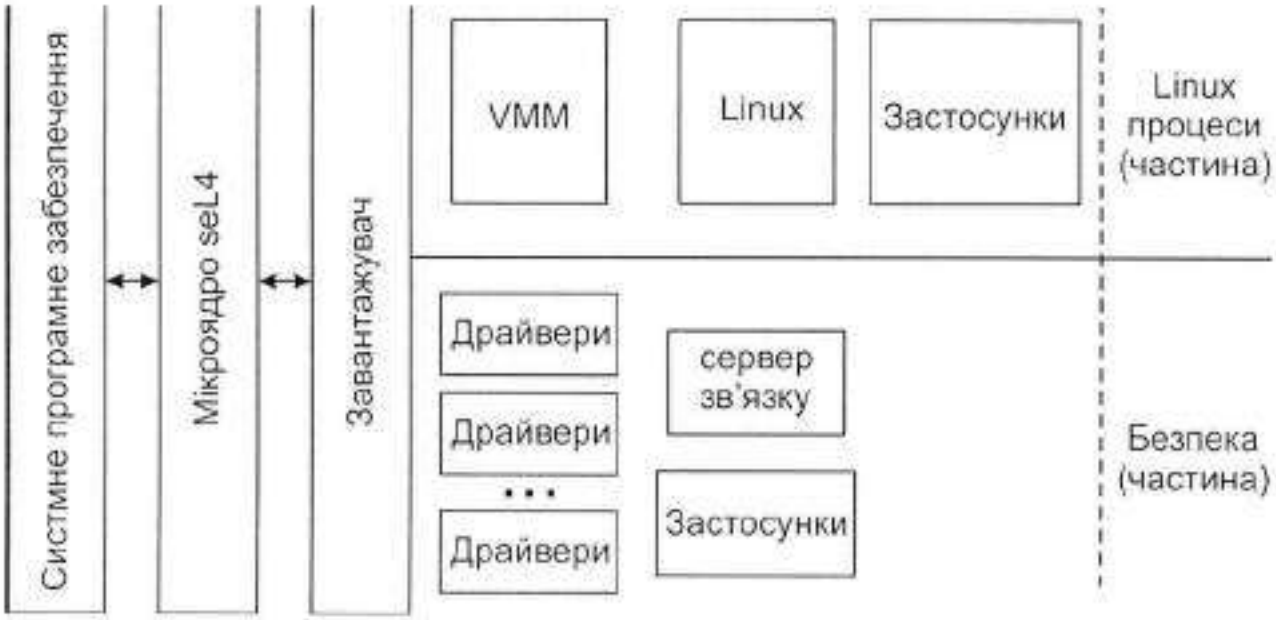


Рисунок 4.1 – Розроблене безпечне середовище виконання з використанням мікровізора seL4

VMM стежить за системними подіями та перериваннями для гостьової ОС, а також перехоплює інструкції привілеїв, щоб передати їх системі seL4. Під час завантаження VMM ініціалізує віртуальний процесор, який налаштовує пряме відображення пам'яті та проходить через апаратне забезпечення. Після налаштування середовища VMM завантажує ядро гостьової ОС, дерево пристроїв і кореневу файлову систему у пам'ять і передає керування ядру. Під час виконання VMM керує можливостями гостьової ОС, просторами віртуальної пам'яті, перериває та зберігає/відновлює стани центрального процесора для контекстних перемикачів тощо.

Для гостьової ОС Linux система запускає налаштований звичайний Linux, який перехресно компілюється для цільової дошки. Ядро Linux та образ побудовані за допомогою вбудованої системи збірки Linux, одного з популярних

інструментів, які автоматизують процес. Linux містить модуль ядра, який при завантаженні в систему зареєструється як віртуальний пристрій, який може взаємодіяти з застосунками Linux через системний виклик.

Останнім компонентом системи є сервер зв'язку. Сервер зв'язку є рідним процесом seL4. Метою цього процесу є обмін даними між процесами Linux, які виконуються у віртуальному середовищі, та застосунками. Сервер зв'язку також є компонентом. Система налаштовує спільну сторінку пам'яті між віртуальною машиною та сервером зв'язку. Для кожної пари зв'язків між сервером зв'язку та іншими процесами seL4 буде використовуватися виділена кінцева точка IPC. Сервер зв'язку виконує функцію диспетчера. Коли повідомлення отримано або зі спільної пам'яті, або з будь-якої кінцевої точки, повідомлення буде поставлено в чергу та передано відповідному одержувачу.

Незважаючи на те, що рішення віртуалізації добре підходить для розділення застарілих і потенційно вразливих застосунків, воно не заважає зловмисникам фізично втручатися в роботу пристрою або змінювати процес завантаження для завантаження шкідливої ОС. Контрзаходом є надійне завантаження. Безпечне завантаження є критично важливою функцією безпеки, яка гарантує, що з пристрою можна завантажити лише авторизоване програмне забезпечення з цифровим підписом. Безпечне завантаження пов'язує програмне забезпечення з базовим обладнанням, застосовуючи криптографічні методи, такі як цифрові підписи, на кожному етапі процесу завантаження системи, починаючи з моменту ініціалізації обладнання. Оскільки лише двійковий файл із належним підписом може бути перевірений і завантажений у систему, безпечне завантаження гарантує, що лише авторизоване програмне забезпечення може виконуватися на певному обладнанні. Використовуючи це рішення, програмне забезпечення може делегувати довіру через так званий «ланцюжок довіри», починаючи від завантажувача і закінчуючи застосунками, призначеними для користувача. Існують різні стандартні рішення для безпечного завантаження, які пропонують як специфікацію, так і реалізацію,

перевірене завантаження тощо. У експерименті плата використовує завантажувач. Реалізовано процедуру безпечного завантаження, яка дозволяє кожному компоненту в системі перевіряти наступний компонент під час процесу завантаження та включати його з перевіреним завантаженням. Процес завантаження seL4 на ARM схожий на процес завантаження Linux. Для конкретної плати розробки процедура завантаження розпочинається при включенні. Процес завантаження відбувається в наступному порядку. Завантажувальний ПЗУ ініціалізує оперативну пам'ять, завантажує вторинний завантажувач у пам'ять і передає керування вторинному завантажувачу з відкритим вихідним кодом для вбудованих пристроїв. Завантажувач зчитує змінні середовища з завантажувального пристрою, виявляє апаратні ресурси, налаштовує процесор у режим і завантажує ядро seL4 та процес кореневого завдання в оперативну пам'ять, перш ніж передати керування завантажувачу. Завантажувач переміщує ядро seL4 і кореневе завдання в потрібну адресу фізичної пам'яті і налаштовує завантажувальні змінні для seL4. Потім він передає управління seL4. Він налаштовує апаратні ресурси та планує кореневе завдання в режимі користувача, перш ніж зробити доступним для запуску. Далі завантажує інші застосунки в режимі користувача і розподіляє ресурси у вигляді можливостей. У цьому конкретному прототипі цими процесами є сервер зв'язку та монітор віртуальної машини. VMM налаштовує віртуальне середовище виконання для гостьової ОС, завантажує двійковий файл ядра Linux і дерево пристроїв у пам'ять та передає керування ядру Linux.

Після процесу завантаження реалізовано процедуру безпечного завантаження, починаючи в системі seL4. Для цього вставлено відкритий ключ і використано закритий ключ з підписом seL4 ядра і завантажувач, а також додано їх підписи в кінці кожного двійкового файла. Перед завантаженням завантажувач обчислює хеш кожного двійкового файла та звіряє цифровий підпис зі значенням хешу. Завантажувач продовжить роботу лише в тому випадку, якщо підпис збігається. Потім включено аналогічну процедуру в

процес ініціалізації завантажувача, щоб переконатися, що завантажувач перевірить сервер зв'язку та VMM. Нарешті, у VMM додано процедуру перевірки, щоб на платі виконувалися лише підписані ядро Linux та об'єкт дерева пристроїв.

Крім того, інтегровано це з перевіреним завантаженням завантажувач. Завантажувач реалізовано з технологією безпечного завантаження, яка дозволяє розробникам вбудовувати криптографічні ключі в завантажувач з метою перевірки. Крім того, перевірене завантаження використовує новий формат файлу, який називається зведеним деревом зображень. Це гнучкий, структурований формат контейнера, який підтримує кілька образів ядра, дерева пристроїв. За допомогою цього формату можна легко додавати підписи та інші метадані, щоб вказати завантажувачу, як завантажити систему. Скомпільовано seL4 та його компоненти в образ, щоб використовувати цей механізм.

У виробництві багато постачальників підтримують рішення безпечного завантаження, сумісні зі специфікацією безпечного завантаження. Вони часто надають завантажувальні ПЗУ з підтримкою безпечного завантаження, а також додаткове апаратне сховище для програмування ключа шифрування, що не підлягає очищенню. Працюючи з ними, можна вбудовувати криптографічний ключ на платі в майбутньому. Такий ключ може бути використаний для підпису вторинного завантажувача, щоб дозволити лише авторизованому завантажувачу з потрібною конфігурацією бути виконаним на платі для наскрізної гарантії.

Щоб оцінити безпеку та практичність цього рішення в домені САБ, прототип прийняв сценарій шлюзу IoT від бізнесу і реалізований у системі.

Одним із критично важливих компонентів безпеки в САБ наступного покоління є шлюз IoT. У той час як багато пристроїв САБ замінюються їх останніми аналогами для забезпечення більш «інтелектуального» управління, пристрої все більше і більше покладаються на хмарні обчислювальні служби. Шлюз IoT — це пристрій, який збирає та зберігає дані з предметного поля і забезпечує зв'язок між локальними системами та хмарою. Як периферійний

пристрій, який служить точкою входу в об'єднання Інтернету, домашніх і промислових систем автоматизації будівель, шлюз IoT відіграє ключову роль у традиційних САБ та Інтернету послуг. Отже, його безпека нерозривно пов'язана з безпекою та захищеністю об'єкта.

Серед усіх функцій шлюзу IoT найбільшою проблемою для бізнесу є захист ключа автентифікації, який використовується для автентифікації, і забезпечення безпечного зв'язку між шлюзом і хмарою. Прагнучи продемонструвати переваги безпеки запропонованого рішення в домені САБ, цей експеримент використовує управління ключами як приклад для оцінки переваг безпеки рішення мікровізора.

У звичайних системах управління ключами покладається на файлову систему і контроль доступу для захисту криптографічних ключів. Під час виконання сервер керування ключами завантажуватиме ключ у свій простір. Сервер керування ключами відповідає за надання криптографічних послуг. Наприклад, шифрування, дешифрування, цифрового підпису, перевірки підпису тощо для авторизованого однорангового процесу. Наприклад хмарного клієнта. За потреби хмарний клієнт зв'яжеться із сервером керування ключами за допомогою сокета локального домену UNIX. Для захисту цих криптографічних ключів від шкідливого доступу, у цій реалізації переміщено керування ключами з частини Linux до розділу seL4. Під час завантаження ключі доступні тільки в розділі seL4 і завантажуються на сервері управління ключами. Хмарний клієнт розміщується в розділі Linux. Коли хмарному клієнту потрібні криптографічні послуги, хмарний клієнт здійснює системний виклик, наприклад, запитує службу ядра або драйверів. Модуль ядра перевірить привілеї процесу. Якщо запит буде задоволено, модуль ядра перенаправить запит на сервер зв'язку в розділі seL4, викликавши відповідну можливість за допомогою примітиву seL4 IPC. Зрештою, зв'язок передавався на сервер керування ключами, а результат повертався до хмарного клієнта за аналогічним шляхом. З точки зору програми Linux, фактична реалізація є прозорою, отже, потрібно дуже мало змін у

вихідному коді.

Щоб оцінити цей підхід, проаналізовано можливі вектори атак у порівнянні зі звичайною реалізацією. Ґрунтуючись на існуючих реальних атаках, модель загроз передбачає атаки двох рівнів. По-перше, атаки віддалено з можливими скомпрометованими процесами. По-друге, атака з фізичним доступом пристрою. Завдяки гарантії функціональної коректності, що надається формальною верифікацією seL4 передбачаєтья, що мікровізор має високу надійність і не містить вразливостей програмного забезпечення.

Враховуючи, що шлюз IoT є периферійним пристроєм у звичайній системі, якщо в одній із програм, запущених у розділі Linux, є вразливості, зловмисники можуть отримати віддалений доступ, використовуючи такі слабкі місця. Після того, як зловмисники отримають доступ до пристрою, зловмисники зможуть отримати доступ до криптографічних ключів безпосередньо з файлової системи, якщо дозвіл на файл налаштовано неправильно або шляхом подальшого підвищення привілеїв. Однак ця загроза пом'якшується рішенням мікровізора. Оскільки криптографічні ключі та сервер керування ключами розміщені у власному розділі seL4, доки фундаментальний IPC та керування пам'яттю в seL4 не зламано, навіть зловмисники з кореневими привілеями Linux не можуть отримати ключі.

У звичайних системах, якщо їх не налаштовано належним чином, зловмисний процес може підробити локальний IPC і зловживати сервером керування ключами без отримання ключів. Цей вид атаки, хоча і можливий, простіше пом'якшити в розчині мікровізора. У рішенні мікровізора, якщо шкідлива програма хоче запитати послуги від сервера керування ключами, їй потрібно буде викликати налаштований системний виклик. Модуль ядра, який обробляє системний виклик, діє як проксі-сервер і може ретельно перевіряти запит. Оскільки модуль працює в просторі ядра з високими привілеями, розробники можуть реалізувати модуль для перевірки ідентичності процесу, що викликає. До тих пір, поки ядро не скомпрометовано, а модуль ядра вставляє

належний захист перевірити треба, перш ніж авторизувати будь-який запит, бо зловмисники не можуть зловживати послугами. Якщо зловмисникам вдасться скомпрометувати ядро Linux, то вони можуть підробити запит і зловживати службами так, ніби вони є власниками ключа. Однак, якщо припустити, що зловмисники можуть скомпрометувати ядро Linux, вони, по суті, заволодіють системою і можуть робити плановані дії. Таким чином, у порівнянні з цим, рішення мікровізора все ще є кращим з точки зору безпеки системи.

З іншого боку, якщо зловмисники мають фізичний доступ до пристрою, потрібно враховувати більше векторів атаки. У звичайній системі зловмисники можуть замінити завантажувальний образ і запустити шкідливу версію, за допомогою якої вони матимуть повний контроль над доступом до сховища та отриманням ключів. За підтримки безпечного завантаження такі зловмисники можуть бути усунені, оскільки непідписані образи ядра та програми не можуть бути завантажені. Для того, щоб повністю захистити систему від фізичного втручання, в обох цих рішеннях повинні застосовуватися додаткові механізми безпеки, такі як шифрування всього диска, апаратні криптографічні чіпи, наприклад, модуль довірчої платформи. Однак, такі механізми простіше застосувати в рідному розділі seL4 через модульну конструкцію в seL4, а також через те, що в такій системі немає застарілої залежності.

Окрім аналізу безпеки, також розглянемо оцінку продуктивності. Доведено, що мікроядро seL4 є найшвидшим мікроядром у світі. Вого на архітектурі ARM знаходиться в межах пари сотень наносекунд для відправки одного повідомлення мінімальної довжини між двома потоками.

Щоб оцінити продуктивність віртуалізації, порівняно продуктивність системи Linux і віртуалізованого Linux за допомогою широко використовуваного інструменту порівняння продуктивності системи. Це набір мікро-бенчмарків, який призначений для вимірювання поширених вузьких місць продуктивності.

У розширеннях віртуалізації ARM єдиними взаємодіями між віртуальною

машиною та мікровізором є перемикання контексту між двома розділами, входи віртуальної машини, виходи та переривання. Отже, накладні витрати мінімальні. Ця оцінка збігається з нашими результатами бенчмарку, віртуальна машина показує майже рідну продуктивність. Багато мікротестів навіть показують негативні накладні витрати, такі як запис, затримка каналу та сокет UNIX. Ці результати, ймовірно, пов'язані з тим, що віртуальна машина використовувала сховище з відображенням пам'яті, а не сховище або конфлікти, пов'язані з кешем.

Таким чином, в середньому деградація віртуальної машини може становити лише 4%. Додаткові витрати на віртуалізацію значною мірою залежать від робочого навантаження та апаратної платформи.

На практиці існує кілька обмежень такого підходу. По-перше, обов'язковий контроль доступу за допомогою матриці контролю доступу вимагає модифікації ядра кожного разу, коли вводяться нові політики. Частково це пов'язано з конструкцією примітивів IPC ядра MINIX 3 та структурою мікроядра. Наприклад, одним із перспективних рішень є автоматична генерація матриці контролю доступу, проксі-процесу та архітектури коду програми з моделі системи. Модель seL4, заснована на можливостях, демонструє ще один перспективний підхід. Подавши заявку на доступ на основі можливостей у ядрі, політику можна впровадити у користувачів, яка є більш гнучкою. Потрібно більше роботи з портування існуючих драйверів та впровадження бібліотек, щоб зробити систему простішою у використанні, а також оптимізувати продуктивність та ефективність. Дослідження ефективності IPC для Linux і MINIX 3 подано в табл. 4.1.

Ця оцінка моделює робоче навантаження системи автоматизації будівлі. Для порівняння, продуктивність мережі Linux вимірюється як базова. Результати подані є середнім арифметичним для збільшених тестів. MINIX IPC працює швидше, ніж Linux IPC при передачі невеликих фрагментів даних. MINX 3 IPC обмежений максимум 64 байтами. Однак більші передачі найбільш оптимальні через спільну пам'ять. Для мережевого зв'язку за допомогою UDP MINIX 3

працює повільніше. Це пов'язано з відсутністю загальної оптимізації продуктивності в MINIX 3 і неефективним мережевим драйвером. Однак така продуктивність все ще прийнятна в домені САБ. Для затримки мережі проксі-сервер додає лише близько 4% додаткових витрат без тунелювання та приблизно від 10% до 30% додаткових витрат при тунелюванні подібно до Linux.

Таблиця 4.1 - IPC для Linux і MINIX 3

Розмір (байти)	Локальний IPC (мікросекунди)		
	Черга повідомлень Linux	Сокет Linux Unix	MINIX 3 IPC
1	20.5	59.6	14
64	23.4	59.3	14
128	25.2	60.4	-
256	27.3	60.7	-
512	25.1	63.2	-
1024	26.6	62.5	-

Ця оцінка не є еталоном для систематичного порівняння продуктивності ядра різних ОС. Однак отриманий результат дає попередню оцінку підходу для САБ. Запропонована структура зв'язку показує результати з додатковими витратами на продуктивність для загальних переваг безпеки зв'язку. Ґрунтуючись на цих вимірюваннях і зібраному реальному наборі даних САБ, рішення є достатнім для вимог до продуктивності САБ.

Таким чином, розроблено безпечну обчислювальну платформу для системи аутентифікації будівель наступного покоління. У процесі вивчення проблемної області проведено емпіричне дослідження двох реально існуючих систем автоматизації будівель. Проаналізовано різні потенційні ризики для безпеки і запропоновано безпечну обчислювальну платформу на основі мікроядра в спробі вирішити ці проблеми безпеки для САБ наступного

покоління. З огляду на сумісність з існуючими системами, це рішення використовує архітектуру операційної системи мікроядра і застосовує обов'язковий контроль доступу та мережевий зв'язок на основі проксі-сервера для забезпечення дотримання глобальних політик в контролерах автоматизації Це рішення забезпечує зворотню сумісність.

Розроблений метод є надійним завдяки мінімальній функціональності мікроядра в просторі ядра та модульній архітектурі. Він має зворотну сумісність, оскільки застарілі програми можуть бути розміщені у віртуальному середовищі, а архітектуру проксі-сервера захищає протокол мережевого зв'язку САБ за допомогою мережевого тунелювання. Для оцінки запропонованого рішення були реалізовані різні сценарії автоматизації будівель, такі як сценарій лабораторного контролю температури, сценарій управління ключами та відповідні атаки. Ці експерименти показують переваги безпеки запропонованого рішення та відчутну ефективність.

Також, представлений підхід віртуалізації, який використовує формально перевірений seL4 як мікровізор для розміщення застарілих систем САБ. Цей підхід ґрунтується на полегшенні адаптації бізнесу, одночасно збалансовуючи компроміси між витратами та безпекою. Крім того, він використовує формально перевірене мікроядро seL4 для його унікальних переваг математично доведеної гарантії безпеки та прогресу в технології віртуалізації ARM. Цей метод надає два розділи. Один для розміщення застарілого програмного забезпечення з багатою функціональністю. Інший безпосередньо працює поверх середовища високої надійності SEL4 для розміщення критично важливих програм. Розміщення віртуальної машини Linux поверх seL4 дозволяє розробникам спочатку захистити критичні елементи, а решту системи залишити в основному незмінною. Система може розвиватися з часом, оскільки програмні продукти змінюються.

4.3 Висновки до четвертого розділу

Розроблено безпечну обчислювальну платформу для системи аутентифікації будівель наступного покоління за технологією кіберфізичних систем. Це рішення забезпечує зворотню сумісність. Розроблений метод є надійним завдяки мінімальній функціональності мікроядра в просторі ядра та модульній архітектурі. Він має зворотну сумісність, оскільки застарілі програми можуть бути розміщені у віртуальному середовищі, а архітектуру проксі-сервера захищає протокол мережевого зв'язку САБ за допомогою мережевого тунелювання. Також, представлений підхід віртуалізації, який використовує формально перевірений seL4 як мікровізор для розміщення застарілих систем САБ. Цей підхід грунтується на полегшенні адаптації бізнесу, одночасно збалансовуючи компроміси між витратами та безпекою. Розміщення віртуальної машини Linux поверх seL4 дозволяє розробникам спочатку захистити критичні елементи, а решту системи залишити в основному незмінною. Система може розвиватися з часом, оскільки програмні продукти змінюються.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено новий метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра та отримано такі результати.

1. Проаналізовано відомі методи автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

2. Розроблено новий метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра. Розроблений метод є надійним завдяки мінімальній функціональності мікроядра в просторі ядра та модульній архітектурі. Він має зворотну сумісність, оскільки застарілі програми можуть бути розміщені у віртуальному середовищі, а архітектуру проксі-сервера захищає протокол мережевого зв'язку САБ за допомогою мережевого тунелювання. Також, представлений підхід віртуалізації, який використовує формально перевірений seL4 як мікровізор для розміщення застарілих систем САБ. Розміщення віртуальної машини Linux поверх seL4 дозволяє розробникам спочатку захистити критичні елементи, а решту системи залишити в основному незмінною.

3. Здійснено реалізацію розробленого методу створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра. Система може розвиватися з часом, оскільки програмні продукти змінюються.

4. Здійснено есперименталні дослідження згідно розроблених рішень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Langner R. To kill a centrifuge: a technical analysis of what stuxnet's creators tried toachieve. *Arlington, VA: Langner Group*, 2013.

2. Karnouskos S. Stuxnet Worm Impact on Industrial Cyber-Physical System Security. In *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2011.

3. Mathews L. Hackers Use DDoS Attack To Cut Heat To Apartments, Nov 2016.

4. Rios B. Owning a Building: Exploiting Access Control and Facility Management Systems. *Black Hat Asia 2014, Singapore*, Mar 2014.

5. McCarty B. *Selinux: NSA's Open Source Security Enhanced Linux*, volume 238. O'Reilly, 2005.

6. Shabtai A., Fledel Y., Elovici Y. Securing android-powered mobile devices usingselinux. *IEEE Security & Privacy*, 8(3), 2010. P. 36–44.

7. Jaeger T. Operating system security. *Synthesis Lectures on Information Security, Privacy and Trust*, 2008.

8. Klein G., Tuch H. Towards verified virtual memory in l4. *TPHOLs Emerging Trends*, 4:16, 2004.

9. Gu R., Shao Z., Chen H., Wu X., Kim J., Sjöberg V., Costanzo D. Certikos: An extensible architecture for building certified concurrent os kernels. In *OSDI*, 2016. Volume 16, P. 653–669,

10. Steinberg U., Kauer B. Nova: a microhypervisor-based secure virtualization architecture. In *Proceedings of the 5th European conference on Computer systems*, ACM, 2010. P. 209–222.

11. Lackorzynski A., Warg A. Timing aware hardware virtualization on the l4re microkernel system. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016. P. 67.

12. Heiser G., Leslie B. The OKL4 Microvisor: Convergence Point of

Microkernels and Hypervisors. In *Proceedings of the 1st ACM APSys*. ACM, 2010.

13. Wang X., Mizuno M., Neilsen M., Ou X., Rajagopalan S. R., Boldwin W. G., Phillips B. Secure RTOS Architecture for Building Automation. *Proceedingsof the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy - CPS-SPC15*, Oct 2015.

14. Koblentz G. D. Biosecurity reconsidered: Calibrating biological threats and responses. *International security*, 2010.

15. U.S. Department of Health, Centers for Disease Control Human Services, Public Health Service, and National Institutes of Health Prevention. *Biosafety in Microbiological and Biomedical Laboratories, 5th Edition*. HHS, 2009.

16. Allen J. G, MacNaughton P., Satish U., Santanam S., Vallarino J., Spengler J. D. Associations of cognitive function scores with carbon dioxide, ventilation, and volatile organic compound exposures in office workers: A controlled exposure study of green and conventional office environments. *NIH*, 2015.

17. Loy D., Dietrich D., Schweinzer H.-J. *Open control networks: LonWorks/EIA 709 technology*. Springer Science & Business Media, 2012.

18. Merz H., Hansemann T., Hübner C. *Building Automation: Communi- cation Systems with EIB/KNX, LON and BACnet*. Springer Science & Business Media, 2009.

19. Carcano A., Coletta A., Guglielmi M., Masera M. I Nai Fovino, and Alberto Trombetta. A Multidimensional Critical State Analysis for Detecting Intrusions in SCADA Systems. *IEEE Transactions on Industrial Informatics*, 2011.

20. Raza S., Wallgren L., Voigt T. SVELTE: Real-time Intrusion Detection in the Internet of Things. *Ad hoc networks*, 2013.

21. Ostrand T. J., Weyuker E. J. The distribution of faults in a large industrialsoftware system. In *ACM SIGSOFT Software Engineering Notes*, ACM, 2002. Volume 27, P. 55–64.

22. Van der Wiel J., Zykov K., Diaz V., Namestnikov Y. Hajime, the Mysterious Evolving Botnet, Apr 2017.

23. Yeh T., Chiu D., Lu K. Persirai: New Internet of Things (IoT) Botnet TargetsIP Cameras, May 2017.

24. Peraković D., Periša M., Cvitić I. Analysis of the IoT Impact on Volume of DDoS Attacks. In *XXXIII Simpozijum o novim tehnologijama u poštanskom i telekomunika- cionom saobraćaju*, 2015.

25. Wang X., Mizuno M., Neilsen M., Ou X., Rajagopalan S. R., Habeeb R., Amaravadi S., Hatcliff J., Varadarajan S. Enhanced Security of Building Automation Systems Through Microkernel-Based Controller Platforms. *Proceedings of the Second Workshop on Communication, Computing, and Networking in Cyber Physical Systems (CCNCPS 2017)*, Jun 2017.

26. Nordholz J. C., Seifert J.-P. Efficient virtualization on hardware with limited virtualization support, 2011.

27. Härtig H., Roitzsch M. Ten years of research on L4-based real-time systems.In *Proceedings of the 8th Real-Time Linux Workshop*, 2006.

28. Tanenbaum A. S., Herder J. N., Bos H. Can We Make Operating Systems Reliable and Secure? Computer, 2006.

29. Klein G., Derrin P., Elphinstone K. Experience report: sel4: Formally verifying a high-performance microkernel. In ACM Sigplan Notices. ACM, 2009.

30. Elphinstone K., Heiser G. From L3 to seL4 what have we learnt in 20 years of L4 microkernels? Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles - *SOSP 13*, 2013.

31. Klein G., Norrish M., Sewell T., Tuch H., Winwood S., Elphinstone K., Heiser G., Andronick J., David Cock, Philip Derrin, and et al. sel4: formal verification of an os kernel. *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, Oct 2009. P. 207–220,

32. Kuz I., Klein G., Lewis C., Walker A. A language for describing capability-based systems. In *Proceedings of the first ACM asia-pacific workshop on Workshopon*

systems, ACM, 2010. P. 31–36.

33. Kuz I., Liu Y., Gorton I., Heiser G. Camkes: A component model for secure microkernel-based embedded systems. *Journal of Systems and Software*, 2007. V. 80(5), P.687–699,

34. Cromer D. C., Freeman J. W., Goodman S. D., Kern E. R., Springfield R. S. Method and system for tracking a secure boot in a trusted computing environment, March 13 2007. US Patent 7,191,464.

35. Verified Boot – Introduction to U-Boot's Secure Boot, Oct 2018.

36. Blackham B., Heiser G. Correct, Fast, Maintainable: Choose Any Three! In Proceedings of the *APSys*. ACM, 2012.

37. Habeeb R. Improving the security of building automation systems through an sel4-based communication framework. Master's thesis, University of South Florida, 2018.

38. Heiser G. The role of virtualization in embedded systems. In *Proceedings of the 1st workshop on Isolation and integration in embedded systems*, ACM, 2008. P. 11–16.

39. Iqbal A., Sadeque N., Mutia R. I. An overview of microkernel, hypervisor and microvisor virtualization approaches for embedded systems. *Report, Department of Electrical and Information Technology, Lund University, Sweden*, 2110:15, 2009.

40. Anderson T., Peterson L., Shenker S., Turner J. Overcoming the internet impasse through virtualization. *Computer*, 2005. V. 38(4), P. 34–41.

41. Acharya A., Buford J., Krishnaswamy V. Phone virtualization using a micro- kernel hypervisor. In *Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference on*, IEEE, 2009. P. 1–6.

42. Varanasi P., Heiser G. Hardware-supported virtualization on arm. In *Proceed- ings of the Second Asia-Pacific Workshop on Systems*, ACM, 2011. P. 11.

43. Adams Keith, Agesen O. A comparison of software and hardware

techniques for x86 virtualization. *ACM SIGARCH Computer Architecture News*, 2006. V. 34(5), P. 2–13.

44. Landis J., Powderly T., Subrahmanian R., Puthiyaparambil A., Hunter J. Computer system para-virtualization using a hypervisor that is implemented in a partition of the host system, July 19 2011. US Patent 7,984,108.

45. Hwang J.-Y., Suh S.-B., Heo S.-K., Park C.-J., Ryu J.-M., Park S.-Y., Kim C.-R. Xen on arm: System virtualization using xen hypervisor for arm- based secure mobile phones. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, IEEE, 2008. P. 257–261.

46. Lackorzynski A. et al. L4linux porting optimizations. *Master's thesis, TU Dresden*, 2004.

47. ARM Cortex. A15 mpcore processor technical reference manual. *ARM Limited, Jun*, 24:12, 2013. Irfan Habib. Virtualization with kvm. *Linux Journal*, 2008(166):8, 2008.

48. Elhage N. Virtunoid: A kvm guest! host privilege escalation exploit. *Black Hat USA*, 2011, 2011.

49. Biggs S., Lee D., Heiser G. The jury is in: Monolithic os design is flawed. *Proceedings of the 9th Asia-Pacific Workshop on Systems*, 2018.

50. Heiser G., Sueur E. L., Danis A., Budzynowski A., Salomie T.-I., Alonso G. Rapilog: Reducing system complexity through verification. In *Proceedings of the 8th ACM European Conference on Computer Systems*, ACM, 2013. P. 323–336.

51. Rios B. Owning a building: Exploit access control and facility management systems. *Black Hat Asia*, 2014.

52. Leverett E. P. *Quantitatively Assessing and Visualising Industrial System Attack Sur- faces*. PhD thesis, University of Cambridge, 2011.

53. Cherbov G., Bolshev A. Icscorsair: How i will pwn your erp through 4-20 ma currentloop. *Black Hat USA*, 2014.

54. Molina J. Learn How to Control Every Room at A Luxury Hotel

Remotely: The Dangers of Insecure Home Automation Deployment. *Black Hat USA*, 2014.

55. Zetter K. *Countdown to Zero Day: Stuxnet and the Launch of the World's First Digital Weapon*. Crown Publishing Group, New York, NY, USA, 2014.

56. Zetter K. Researchers Hack Building Control System at Google Australia Office, May 2013.

57. Mirsky Y., Guri M., Elovici Y. Hvacker: Bridging the air-gap by attackingthe air conditioning system. *arXiv preprint arXiv:1703.10454*, 2017.

58. Booth S., Barnett J., Burman K., Hambrick J., Westby R. *Net zero energy military installations: A guide to assessment and planning*. National Renewable Energy Laboratory, 2010.

59. Granzer W. *Secure communication in home and building automation systems*. na, 2010.

60. Granzer W., Praus F., Kastner W. Security in Building Automation Systems. *IEEE Transactions on Industrial Electronics*, 2010.

61. Granzer W., Lechner D., Praus F. Securing IP Backbones in Building Automation Networks. *2009 7th IEEE International Conference on Industrial Informatics*, Jun 2009.

62. Dünhaupt S. Vulnerabilities of industrial automation systems. Master's thesis, RUHR- Universität bochum, 2012.

63. Praus F., Kastner W., Palensky P. Software Security Requirements in Building Automation. *Sicherheit 2016*, 2016.

64. Feng M., Xiao B., Yu B., Qian J., Zhang X., Chen P., Li, B. A Novel Deception Defense-Based Honeypot System for Power Grid Network. *International Conference on Smart Computing and Communication*, 2021, Vol. 13202, pp. 297-307. Cham: Springer International Publishing. DOI: 10.1007/978-3-030-97774-0_27.

65. Walter E., Ferguson-Walter K., Ridley A. Incorporating deception into cyberbattlesim for autonomous defense. 2021. *arXiv preprint arXiv:2108.13980*. DOI: 10.48550/arXiv.2108.13980.

66. Anwar A. H., Kamhoua C. A., Leslie N. O., Kiekintveld C. Honeypot Allocation for Cyber Deception Under Uncertainty. *IEEE Transactions on Network and Service Management,* 2022, vol. 19. no. 3, pp. 3438-3452. DOI: 10.1109/TNSM.2022.3179965.

67. Sayed M. A., Anwar A. H., Kiekintveld C., Kamhoua C. Honeypot Allocation for Cyber Deception in Dynamic Tactical Networks: A Game Theoretic Approach. *14th International Conference on Decision and Game Theory for Security. GameSec 2023.* 2023. arXiv preprint. arXiv:2308.11817. DOI: 10.48550/arXiv.2308.11817.

68. Anwar A. H., Kamhoua C. A. Cyber Deception using Honeypot Allocation and Diversity: A Game Theoretic Approach. *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC),* Las Vegas, NV, USA, 2022, pp. 543-549. DOI: 10.1109/CCNC49033.2022.9700616.

69. Anwar A. H., Kamhoua C., Leslie N. Honeypot allocation over attack graphs in cyber deception games. *International Conference on Computing, Networking and Communications (ICNC),* 2020, pp. 502-506, IEEE. DOI: 10.1109/ICNC47757.2020.9049764.

70. Acosta J. C., Basak A., Kiekintveld C., Kamhoua C. Lightweight On-Demand Honeypot Deployment for Cyber Deception. In Gladyshev, P., Goel, S., James, J., Markowsky, G., Johnson, D. (eds) Digital Forensics and Cyber Crime. ICDF2C 2021. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering,* 2022, vol. 441, pp. 294-312. Springer, Cham. DOI: 10.1007/978-3-031-06365-7_18.

71. Priya D., Chakkaravarthy S. Containerized cloud-based honeypot deception for tracking attackers. *Scientific Reports,* 2023, vol. 13. DOI: 10.1038/s41598-023-28613-0.

72. Al-Shaer E., Wei J., Hamlen K. W., Wang C. Autonomous Cyber Deception. Reasoning. Adaptive Planning. and Evaluation of HoneyThings. Springer Nature Switzerland AG, 2019. DOI: 10.1007/978-3-030-02110-8.

73. Wegerer M., Tjoa S. Defeating the Database Adversary Using Deception – A MySQL Database Honeypot. *International Conference on Software Security and Assurance (ICSSA),* Saint Pölten. Austria, 2016. pp. 6-10. DOI: 10.1109/ICSSA.2016.8.

74. Kedrowitsch A., Danfeng Y., Gang W., Cameron K. A First Look: Using Linux Containers for Deceptive Honeypots. *Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense (SafeConfig '17).* Association for Computing Machinery, New York, NY, USA, 2017, pp. 15–22. DOI: 10.1145/3140368.3140371.

75. Almeshekah M. H., Spafford E. H. Cyber Security Deception. In: Jajodia. S., Subrahmanian. V., Swarup. V., Wang. C. (eds) *Cyber Deception,* 2016, p. 318, Cham. Springer. DOI: 10.1007/978-3-319-32699-3_2.

76. Zobal L., Kolář D., Fujdiak R. Current State of Honeypots and Deception Strategies in Cybersecurity. *11th International Congress on Ultra-Modern Telecommunications and Control Systems and Workshops (ICUMT).* Dublin. Ireland. 2019. pp. 1-9. DOI: 10.1109/ICUMT48472.2019.8970921.

77. Dahbul R. N., Lim C., Purnama J. Enhancing honeypot deception capability through network service fingerprint. *Journal of Physics: Conference Series,* 2017, vol. 801, article no. 012057. DOI: 10.1088/1742-6596/801/1/012057.

78. Razali M. F., Razali M. N., Mansor F. Z., Muruti G., Jamil N. IoT Honeypot: A Review from Researcher's Perspective. *IEEE Conference on Application. Information and Network Security (AINS).* Langkawi. Malaysia, 2018. pp. 93-98. DOI: 10.1109/AINS.2018.8631494.

79. La Q. D., Quek T. Q. S., Lee J., Zhu H. Deceptive Attack and Defense Game. Honeypot-Enabled Networks for the Internet of Things. *IEEE Internet of Things Journal,* 2016, vol. 3, no. 6. pp. 1025-1035. DOI: 10.1109/JIOT.2016.2547994.

80. Rowe N. C. Honeypot Deception Tactics. In: Al-Shaer, E., Wei, J., Hamlen, K., Wang, C. (eds) *Autonomous Cyber Deception.* Springer. Cham, 2019. DOI: 10.1007/978-3-030-02110-8_3.

81. Мельник В.М., Сорочинський О.Ю., Глухенький О.А., Семенюк Б.В. Метод створення безпечної кіберфізичної системи для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра / Збірник наукових праць за матеріалами XV Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2023». Хмельницький, 2023, С.190-192. https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2023-corpuspaper.pdf

ДОДАТОК А Презентація роботи

2 **Зв'язок роботи з науковими програмами, планами, темами.**

Актуальність роботи полягає в необхідності розробити метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

Дослідження, представлені у кваліфікаційній роботі, проводились в рамках студентської наукової роботи кафедри комп'ютерної інженерії та інформаційних систем Хмельницького національного університету.

3 **Перелік публікацій**

За темою кваліфікаційної роботи опубліковано одну публікацію [81] у Збірнику наукових праць за матеріалами XV Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2023». (Хмельницький – 2023. – С.190-192).

Метою кваліфікаційої роботи є розробка методу створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

Поставлена мета досягається розв'язанням таких основних завдань:

- *проаналізувати відомі методи автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра;*
- *розробити метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра;*
- *реалізувати розроблений метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра;*
- *здійснити експериментальні дослідження згідно розроблених рішень.*

- *Об'єктом дослідження є процес створення кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.*
- *Предметом дослідження є методи створення кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.*

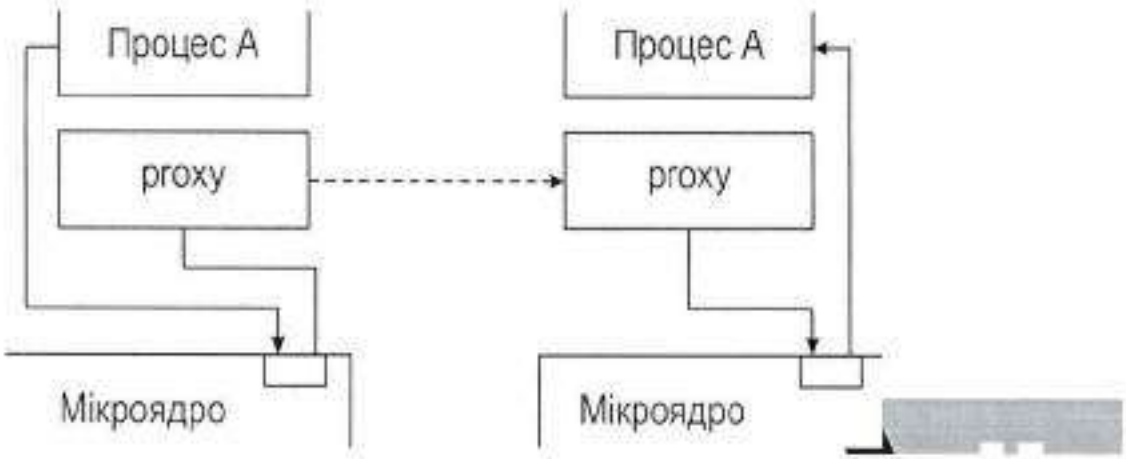
Наукова новизна отриманих результатів:
розроблено новий метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

Практична значимість *отриманих результатів полягає у розробленій крипографічній системі для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра. Вона базується на технології мікроядра для забезпечення безпеки.*

Системи автоматизації будівель на основі кіберфічних систем [1-4] – це великомасштабні розподілені системи керування, які спрямовані на покращення автономного керування та індивідуального управління середовищем будівлі. Типовий САБ контролює кожен аспект роботи будівлі – від освітлення, опалення, вентиляції та кондиціонування повітря до пожежної безпеки, контролю доступу, відео тощо. Операції САБ безпосередньо впливають на енергоспоживання об'єктів будівлі. Тому, зростає мотивація розробляти нові технології для САБ для підвищення комфорту пасажирів і в той же час зниження енергоспоживання та експлуатаційних витрат.

Традиційно САБ складаються з декількох автономних підсистем, специфічних для конкретних застосунків. Кожна підсистема надає лише одну послугу, таку як контроль температури, вентиляція тощо. Пристрої в кожній підсистемі пов'язані між собою через мережі автоматизації будівель, які раніше були відокремлені від IT-систем у навколишньому середовищі. Управління логікою управління здійснюється за допомогою центральних серверів управління, які називаються системами управління будівлею, які з'єднуються з програмованими логічними контролерами (ПЛК) за схемою ведучий-підлеглий. Різні підсистеми об'єднуються в різні домени управління (наприклад, домен безпеки, домен безпеки) і управляються окремо.

Схема обміну даними через віддалені проксі



Кроки методу:

1) управління САБ обробляє видачу повідомлень для ініціювання зв'язку;

2) запит потрапляє в ядро, а ядро перевіряє валідність цього зв'язку з МКД;

3) ядро пересилає повідомлення проксі-серверу, який очікує на повідомлення;

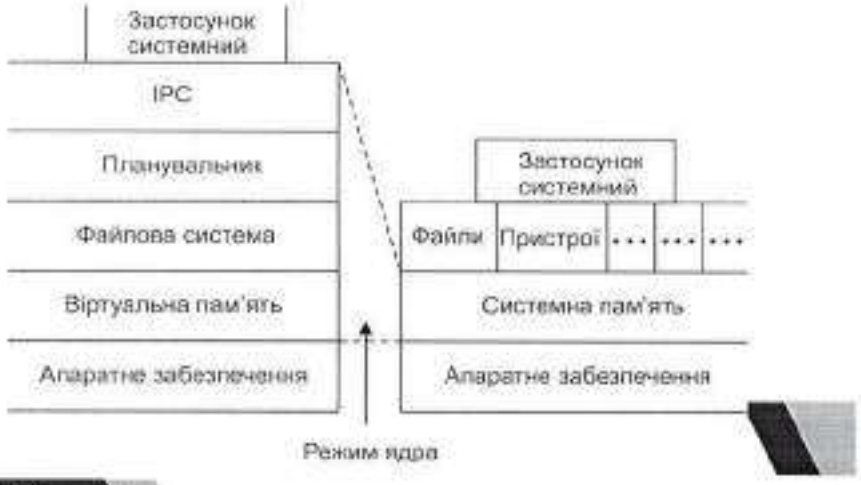
4) проксі-сервер отримує повідомлення через надіслане йому повідомлення-підтвердження;

5) проксі-сервер перевіряє валідність вмісту повідомлень для конкретної програми;

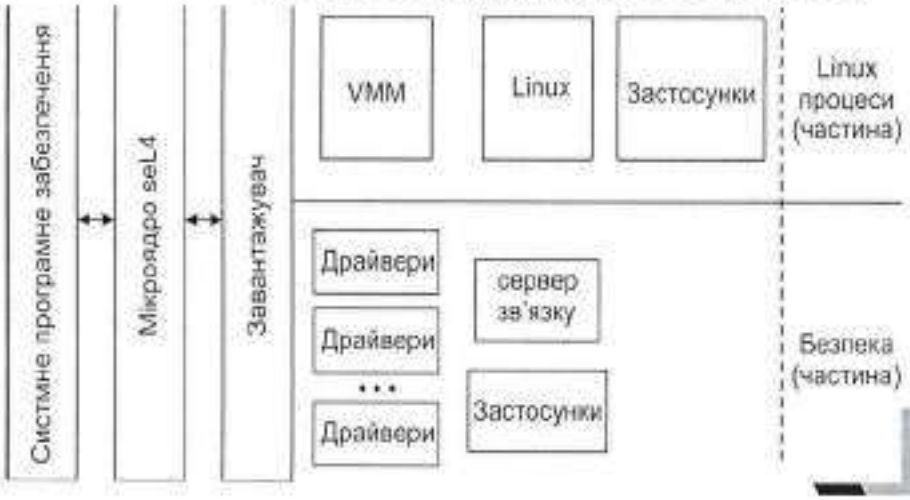
6) проксі-сервер шифрує і пересилає вихідне повідомлення на мережевий сервер;

7) проксі-сервер відповідає на процес керування САБ, вказуючи на те, що повідомлення було надіслано.

Схема режимів ядра ОС



Безпечне середовище виконання з використанням мікровізора seL4



Результати експерименту

Розмір (байти)	Локальний IPC (мікросекунди)		
	Черга повідомлень Linux	Сокет Linux Unix	MINIX 3 IPC
1	20.5	59.6	14
64	23.4	59.3	14
128	25.2	60.4	-
256	27.3	60.7	-
512	25.1	63.2	-
1024	26.6	62.5	-

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено новий метод криптографічного захисту протоколів в засобах комунікації інтернету речей та отримано такі результати:

1. Проаналізовано відомі методи автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра.

2. Розроблено новий метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра. Розроблений метод є надійним завдяки мінімальній функціональності мікроядра в просторі ядра та модульній архітектурі. Він має зворотну сумісність, оскільки застарілі програми можуть бути розміщені у віртуальному середовищі, а архітектуру проксі-сервера захищає протокол мережевого зв'язку САБ за допомогою мережевого тунелювання. Також, представлений підхід віртуалізації, який використовує формально перевірений seL4 як мікровізор для розміщення застарілих систем САБ. Розміщення віртуальної машини Linux поверх seL4 дозволяє розробникам спочатку захистити критичні елементи, а решту системи залишити в основному незмінною.

3. Здійснено реалізацію розробленого методу створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра. Система може розвиватися з часом, оскільки програмні продукти змінюються.

4. Здійснено есперементальні дослідження згідно розроблених рішень.

ДОДАТОК Б Наукова праця здобувача

УДК 004.5

Мельник В.М., Сорочинський О.Ю., Глухенький О.А., Семенюк Б.В.

Хмельницький національний університет

МЕТОД СТВОРЕННЯ БЕЗПЕЧНОЇ КІБЕРФІЗИЧНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ ПРИМІЩЕНЬ ПІДПРИЄМСТВ З ВИКОРИСТАННЯМ ОПЕРАЦІЙНИХ СИСТЕМ НА ОСНОВІ МІКРОЯДРА

Розроблено метод створення безпечної кіберфізичної системи для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра. Реалізовано на системному рівні забезпечення надійних обчислень на основі пристроїв і контролерів. Використовуються такі основні функції безпеки, як надійний модульний дизайн, невеликий код привілеїв і можливість формальної перевірки архітектури мікроядра.

Розроблено концепцію операційної системи з підвищеним рівнем безпеки з вбудованим обов'язковим контролем доступу і комунікаційну основу на основі проксі для контролерів автоматизації приміщень. Це рішення забезпечує комунікацію та ізоляцію між критичними та некритичними програмами в потенційно складному кіберсередовищі.

A method of creating a safe cyber-physical system for the automation of enterprise premises using microkernel-based operating systems has been developed. It is implemented at the system level to ensure reliable calculations based on devices and controllers. Core security features such as robust modular design, small privilege code, and the ability to formally verify the microkernel architecture are used.

A security-enhanced operating system concept with built-in mandatory access control and a proxy-based communication framework for room automation controllers has been developed. This solution provides communication and isolation between critical and non-critical applications in this complex cyber environment.

Системи автоматизації приміщень підприємств – це великомасштабні розподілені системи керування, які мають на меті покращення автономного контролю та індивідуального керування середовищем приміщень. А типова така система контролює всі аспекти роботи будівлі від освітлення, опалення, вентиляції та кондиціонування повітря до протипожежної безпеки, контролю доступу, відео та багато іншого. Операції в ній безпосередньо впливають на споживання енергії будівельними об'єктами. Тому, актуальним є завдання з розробки нової кіберізичної системи для приміщень підприємств, щоб покращити комфорт, зменшити енергоспоживання та експлуатаційні витрати.

Традиційно такі системи [1-5] складаються з кількох автономних підсистем для конкретної програми. Кожна субсистема забезпечує лише одну послугу,

наприклад контроль температури, вентиляцію тощо. Пристрої в кожній підсистемі підключені через мережі автоматизації будівель, які раніше були відділені від IT-систем у середовищі. Логіка управління здійснюється через центральне управління сервером, які називаються системами управління приміщень, які з'єднуються з програмованими логічними контролерами у схемі головний-підлеглий. Різні підсистеми об'єднані в різні контрольні домени (наприклад, домен безпеки, домен безпеки) і керуються окремо.

Зі швидким комерційним розширенням кіберфізичних систем або загальновідомих технологій Інтернету речей у промисловості, розробники та дослідники прагнули охопити концепцію «розумних будинків», яка використовує різні датчики для кращого розуміння контексту життя, надає приводи з підтримкою IP для керування навколишнім середовищем і з'єднує існуючу систему керування з тим, щоб краще реагувати на індивідуальні потреби. Будинки зазнають трансформації краще обслуговувати клієнтів і мешканців за допомогою передової автоматизації та мереж.

Система автоматизації приміщень підприємств – це складна розподілена система керування, яка широко використовується в комерційних, житлових і промислових приміщеннях для моніторингу та керування механічним/електричним обладнанням. Завдяки зростанню промислових і технологічних досягнень, компоненти управління стають все більш взаємопов'язаними. Разом із потенційними перевагами інтеграція також створює нові вектори атак, що значно підвищує ризики безпеки та безпеки в системі керування. Історично склалося так, що система не має дизайну безпеки та покладається на фізичну ізоляцію та «безпеку через невідомість». Ці методи неприйнятні з технологіями «розумного будинку». Галузь потребує переоцінки безпеки та безпеки поточної автоматизації приміщень, систему та розробку комплексного рішення для забезпечення цілісності, надійності та конфіденційності як на рівні системи, так і на рівні мережі.

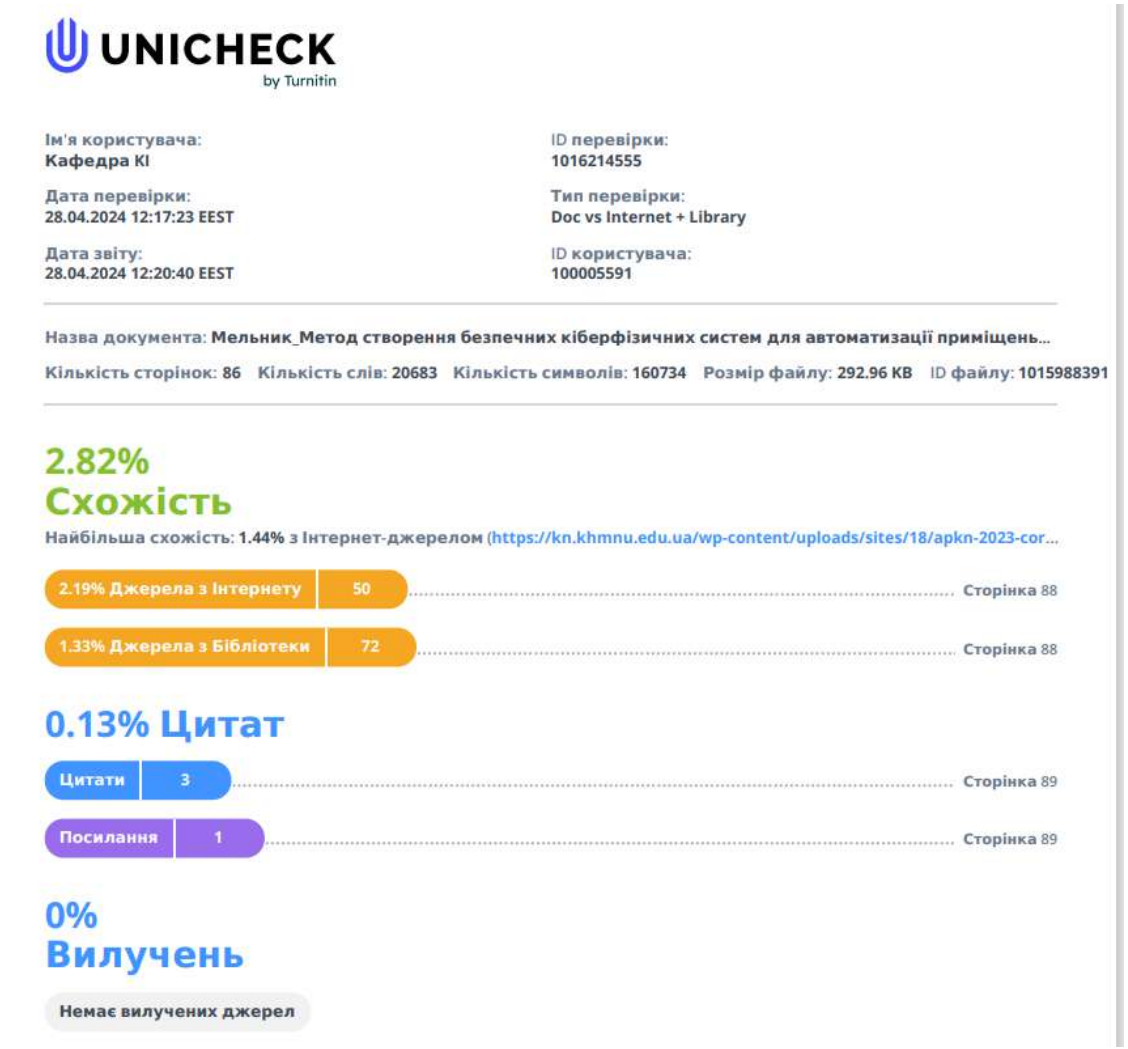
Реалізовано на системному рівні забезпечення надійних обчислень на основі пристроїв і контролерів. Використовуються такі основні функції безпеки, як надійний модульний дизайн, невеликий код привілеїв і можливість формальної перевірки архітектури мікроядра,

Розроблено концепцію операційної системи з підвищеним рівнем безпеки з вбудованим обов'язковим контролем доступу і комунікаційну основу на основі проксі для контролерів автоматизації приміщень. Це рішення забезпечує комунікацію та ізоляцію між критичними та некритичними програмами в потенційно складному кіберсередовищі.

Перелік посилань

1. Friedrich Praus, Wolfgang Kastner, and Peter Palensky. Software Security Requirements in Building Automation. Sicherheit 2016, 2016.
2. Ronghui Gu, Zhong Shao, Hao Chen, Xiongnan (Newman) Wu, Jieung Kim, Vilhelm Sjoberg, and David Costanzo. Certikos: An extensible architecture for building certified concurrent os kernels. In OSDI, volume 16, pages 653–669, 2016.
3. Adam Lackorzynski and Alexander Warg. Timing aware hardware virtualization on the l4remicrokernel system. In 2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), page 67, 2016.
4. Jornt van der Wiel, Konstantin Zykov, Vicente Diaz, and Yury Namestnikov. Hajime, the Mysterious Evolving Botnet, Apr 2017.5. Fukang Liu, Christoph Dobraunig, Florian Mendel, Takanori Isobe, Gaoli Wang, and Zhenfu Cao. Efficient Collision Attack Frameworks for RIPEMD-160. In Alexandra Boldyreva and Daniele Micciancio, editors, Advances in Cryptology – CRYPTO 2019, pages 117–149, Cham, 2019. Springer International Publishing.
5. Xiaolong Wang, Masaaki Mizuno, Mitch Neilsen, Xinming Ou, S. Raj Rajagopalan, Richard Habeeb, Siddharth Amaravadi, John Hatcliff, and Srivatsan Varadarajan. Enhanced Security of Building Automation Systems Through Microkernel-Based Controller Platforms. Proceedings of the Second Workshop on Communication, Computing, and Networking in Cyber Physical Systems (CCNCPS 2017), Jun 2017.

ДОДАТОК В Результати перевірки на антиплагіат



UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016214555

Дата перевірки:
28.04.2024 12:17:23 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
28.04.2024 12:20:40 EEST

ID користувача:
100005591

Назва документа: Мельник_Метод створення безпечних кіберфізичних систем для автоматизації приміщень...

Кількість сторінок: 86 Кількість слів: 20683 Кількість символів: 160734 Розмір файлу: 292.96 KB ID файлу: 1015988391

2.82%
Схожість

Найбільша схожість: 1.44% з Інтернет-джерелом (https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2023-cor...

2.19% Джерела з Інтернету 50 ..... Сторінка 88

1.33% Джерела з Бібліотеки 72 ..... Сторінка 88

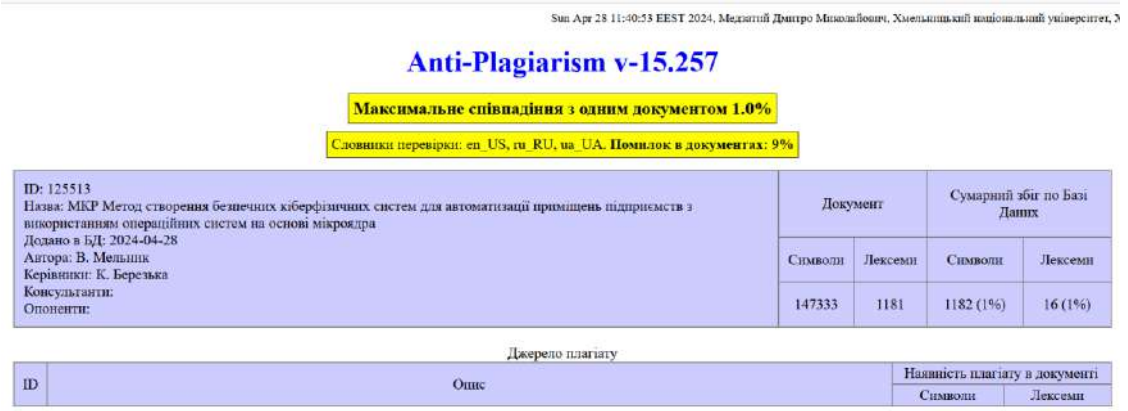
0.13% Цитат

Цитати 3 ..... Сторінка 89

Посилання 1 ..... Сторінка 89

0%
Вилучень

Немає вилучених джерел



Sun Apr 28 11:40:53 EEST 2024, Медзатий Дмитро Миколайович, Хмельницький національний університет, У

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. Помилок в документах: 9%

ID: 125513 Назва: МКР Метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра Додано в БД: 2024-04-28 Автора: В. Мельник Керівники: К. Березька Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	147333	1181	1182 (1%)	16 (1%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

ДОДАТОК Г Заява та висновок про аналіз результатів на антиплагіат

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Тетяні ГОВОРУЩЕНКО

Вікторії МЕЛЬНИК
ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-22-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

31 березня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра

Автор: Мельник Вікторія

Спеціальність: 123 – Компютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Березька Катерина Миколаївна, к.т.н, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-30 джерелами на один фрагмент речення;
2) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 1,33% і адресується до джерел з бібліотеки, що, з урахуванням наведених обґрунтувань, відповідає характеру завдання і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Катерина БЕРЕЗЬКА

Гарант ОП _____ Олег САВЕНКО

Завідувач кафедри КІІС _____ Тетяна ГОВОРУЩЕНКО

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник: Вікторія МЕЛЬНИК

Тема: Метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень - ; кількість сторінок записки 103

1. Короткий зміст роботи та прийнятих рішень У роботі розроблено метод криптографічного захисту протоколів в засобах комунікації інтернету речей

2. Висновок про відповідність роботи дипломному завданню Кваліфікаційна робота відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: у вступі подано об'єкт та предмет дослідження, мету, наукову новизну та практичну цінність роботи, а також характеристику структури роботи. У першому розділі проведено аналіз відомих рішень щодо створення безпечних кіберфізичних систем для автоматизації приміщень підприємств. У другому розділі подано дослідження безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра. У третьому розділі подано розроблений метод створення безпечних кіберфізичних систем для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра. У четвертому розділі здійснено проєктування прототипу з урахуванням зворотньої сумісності та віртуалізації. У висновках підведено підсумки досягнення результатів з розв'язання завдань дослідження.

4. Позитивні сторони роботи:

5. Негативні сторони роботи: немає.

6. Оцінка графічного оформлення та пояснювальної записки роботи: __–____

7. Відгук про роботу в цілому: __Робота виконана на належному рівні.

8. Інші зауваження: ___–_____

9. Оцінка дипломної роботи:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи вважаю, що робота заслуговує оцінки «добре» 4,50 (B)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) __Мартинюк Валерій Володимирович, д.т.н., професор, завідувач кафедри АКІТР ХНУ

"_1_" __травня__2024р. _____