https://doi.org/10.31891/2307-5732-2026-361-79
УДК 004.75

НАГОРНЮК ОЛЕГ
Хмельницький національний університет
https://orcid.org/0009-0000-0887-5520
e-mail: medringone@gmail.com
ДРОЗД АНДРІЙ
Хмельницький національний університет
https://orcid.org/0009-0008-1049-1911
e-mail: andriydrozdit@gmail.com
МЕДЗАТИЙ ДМИТРО
Хмельницький національний університет
https://orcid.org/0009-0004-3247-6406
e-mail: medzatyid@khmnu.edu.ua

МЕТОД ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ТА ЗАХИЩЕНОСТІ КІБЕРФІЗИЧНИХ СИСТЕМ РЕАЛЬНОГО ЧАСУ НА ОСНОВІ БАЛАНСУВАННЯ ЗАВДАНЬ І РЕСУРСІВ

У статті досліджується проблема забезпечення безпеки кіберфізичних систем реального часу (КФС РЧ), що функціонують у середовищах із жорсткими часовими обмеженнями та часто реалізуються мовами програмування без вбудованого захисту пам'яті, такими як C/C++. Традиційні механізми безпеки, призначені для універсальних обчислювальних систем, створюють значні додаткові витрати й обмежують продуктивність, що робить їх застосування у КФС РЧ проблематичним. Основною метою роботи є підвищення рівня захищеності цих систем при мінімальному впливі на часові характеристики виконання завдань. Для досягнення цієї мети запропоновано адаптивні механізми забезпечення цілісності потоків даних, керуючого потоку та вказівників із використанням консервативних оцінок найгіршого часу виконання (WCET) та резервного часу (slack time). Резервний час використовується для динамічного виконання додаткових перевірок без порушення дедлайнів, що дозволяє формально гарантувати своєчасність роботи системи. Потоки даних та керуючий потік формалізовано у вигляді графів із матрицями суміжності та досяжності, що забезпечує точну оцінку відповідності фактичних операцій заздалегідь визначеним політикам безпеки. Для вказівників застосовується контекстно-залежна вибіркова перевірка, що зменшує витрати часу на контроль. Розроблено узагальнену оптимізаційну модель, яка дозволяє балансувати між рівнем безпеки та продуктивністю шляхом використання доступного резерву часу та адаптивного планування завдань. Запропонований покроковий метод включає моделювання системи, аналіз механізмів захисту, оцінку та використання slack time, адаптивне планування завдань, інтеграцію просторової та часової ізоляції, динамічне управління безпекою та тестування із валідацією. Емпіричні дослідження підтверджують ефективність підходу: забезпечується своєчасне виконання критичних завдань, мінімізуються додаткові витрати на перевірки, підвищується рівень захищеності без шкоди продуктивності. Отримані моделі та методи створюють наукову основу для подальшого розвитку адаптивних механізмів безпеки КФС РЧ і їх інтеграції в ресурсоефективні системи реального часу.

***Ключові слова**: кіберфізична система, операційні системи реального часу, керуючі потоки, вказівники, потоки виконання, оптимізація, балансування ресурсів.*

NAHORNIUK OLEH
DROZD ANDRIY
MEDZATYI DMYTRO
Khmelnytskyi National University, Khmelnytskyi, Ukraine

METHOD OF OPTIMIZING THE PRODUCTIVITY AND SECURITY OF REAL-TIME CYBERPHYSICAL SYSTEMS BASED ON BALANCE OF TASKS AND RESOURCES

The paper investigates the problem of ensuring the security of real-time cyber-physical systems (RFCs) that operate in environments with tight time constraints and are often implemented in programming languages without built-in memory protection, such as C/C++. Traditional security mechanisms designed for general-purpose computing systems create significant additional costs and limit performance, which makes their application in RF CFS problematic. The main goal of the work is to increase the level of security of these systems with minimal impact on the time characteristics of task performance. To achieve this goal, adaptive mechanisms for ensuring the integrity of data flows, control flow, and pointers using conservative estimates of worst-case execution time (WCET) and slack time are proposed. Reserve time is used to dynamically perform additional checks without violating deadlines, which allows you to formally guarantee the timely operation of the system. Data flows and control flow are formalized as graphs with adjacency and reachability matrices, which provides an accurate assessment of the compliance of actual operations with predefined security policies. For pointers, a context-dependent selective check is applied, which reduces the time spent on control. A generalized optimization model has been developed, which allows balancing the level of security and productivity by using the available time reserve and adaptive scheduling of tasks. The proposed step-by-step method includes system modeling, analysis of protection mechanisms, evaluation and use of slack time, adaptive task scheduling, integration of spatial and temporal isolation, dynamic security management, and testing with validation. Empirical studies confirm the effectiveness of the approach: it ensures timely execution of critical tasks, minimizes additional costs for inspections, increases the level of security without compromising productivity. The obtained models and methods create a scientific basis for the further development of adaptive safety mechanisms of the RF CFS and their integration into resource-efficient real-time systems.

***Keywords:** cyber-physical system, real-time operating systems, control flows, pointers, execution flows, optimization, resource balancing.*

Стаття надійшла до редакції / Received 17.12.2025 Прийнята до друку / Accepted 11.01.2026 Опубліковано / Published 29.01.2026		This is an Open Access article distributed under the terms of the Creative Commons CC-BY 4.0

Постановка проблеми у загальному вигляді
та її зв'язок із важливими науковими чи практичними завданнями

Кіберфізичні системи реального часу (КФС РЧ) відіграють дедалі важливішу роль у сучасному світі, функціонуючи в середовищах із жорсткими часовими обмеженнями та охоплюючи спектр від простих вбудованих пристроїв, як-от мікроконтролери, до складних платформ, таких як дрони та автономні транспортні засоби. Ці системи переважно програмуються мовами без механізмів захисту пам'яті, наприклад C/C++, що спричиняє численні вразливості безпеки та пов'язані проблеми, а в критичних для безпеки застосуваннях забезпечення надійного захисту є ключовим для гарантії безпеки та дотримання часових вимог. Існуючі механізми безпеки, розроблені для універсальних обчислювальних систем, генерують значні додаткові витрати, що ускладнює їх інтеграцію в КФС РЧ, тоді як модель обчислень у системах реального часу дозволяє впроваджувати захисні механізми з мінімальними витратами ресурсів, тому доцільно розробляти спеціально адаптовані механізми безпеки для РЧ, мінімізуючи їхній вплив на продуктивність. Ця робота розширює три ключові механізми захисту — цілісність потоків даних, керуючого потоку та вказівників, — з адаптаціями для РЧ, що включають перевірку цілісності потоків даних у вільний час під час ітерацій завдань для забезпечення міцного захисту із мінімальними витратами в найгіршому випадку, асинхронну перевірку цілісності керуючого потоку у вбудованих системах через вікна планування для зменшення впливу на реальний час, а також використання доступного системного часу для перевірки цілісності вказівників, що підвищує загальну безпеку без порушення розкладу завдань, таким чином пропонуючи підходи для оптимізації балансу між рівнем безпеки та продуктивністю в системах реального часу.

Актуальність роботи полягає в розробці адаптованих механізмів захисту для кіберфізичних систем реального часу (КФС РЧ), які забезпечують надійну безпеку з мінімальними додатковими витратами на продуктивність, враховуючи жорсткі часові обмеження та вразливості, пов'язані з мовами програмування без захисту пам'яті, такими як C/C++.

Аналіз досліджень та публікацій

Кіберфізичні системи реального часу (КФС РЧ), як ключовий елемент сучасної технологічної інфраструктури, відіграють провідну роль у інтеграції цифрових і фізичних процесів, забезпечуючи взаємодію між обчислювальними компонентами та реальним середовищем. У рамках досліджень, присвячених розвитку інтелектуальних систем, КФС РЧ характеризуються як гібридні структури, де програмне забезпечення генерує команди для фізичних дій, а фізичні параметри, у свою чергу, впливають на обчислювальні процеси, формуючи замкнутий цикл зворотного зв'язку. Цей підхід знаходить широке застосування в різноманітних галузях, від простих вбудованих мікроконтролерів у побутовій техніці до складних систем, таких як медичні імплантати, автономні транспортні засоби, промислові роботи та безпілотні літальні апарати. Дослідження в цій сфері підкреслюють, що на відміну від традиційних обчислювальних платформ, де пріоритетом є максимальна продуктивність, КФС РЧ вимагають суворого дотримання часових обмежень, оскільки будь-які затримки можуть спричинити критичні збої з потенційно катастрофічними наслідками. Для моделювання таких систем наукова спільнота застосовує парадигми реального часу, де навантаження представлено як набір періодичних завдань, кожне з яких асоційоване з жорстким дедлайном [1, 2]. У контексті жорстких дедлайнів, типових для критичних застосувань, забезпечення своєчасності виконання є фундаментальною вимогою, що гарантує не лише функціональну надійність, але й загальну безпеку системи, як це детально аналізується в теоріях планування завдань [3, 4].

У науково-дослідницькому контексті забезпечення безпеки КФС РЧ набуває особливого значення, оскільки ці системи часто інтегруються в критичні інфраструктури [5, 6], де вразливості можуть бути використані для зловмисних атак, загрожуючи людському життю та економічній стабільності. Дослідження підкреслюють, що обмеження за розміром, вагою та енергоспоживанням апаратних платформ зумовлюють використання ресурсоощадних архітектур, які, однак, обмежують обчислювальні потужності. З огляду на вимоги до ефективності, розробка КФС РЧ переважно базується на мовах програмування C та C++, які забезпечують низькорівневий контроль над ресурсами, але водночас позбавлені вбудованих механізмів захисту пам'яті. Це призводить до появи численних вразливостей, пов'язаних із буферними переповненнями, витоками пам'яті та іншими помилками, які зловмисники можуть експлуатувати для несанкціонованого доступу [7, 8]. Процес експлуатації таких атак, як правило, включає етапи виявлення помилок у коді, створення шкідливого навантаження та модифікацію даних у пам'яті процесу, що порушує цілісність виконання програми. Атаки на пошкодження пам'яті можуть призводити до непередбачуваної поведінки системи, компрометуючи як кібер-, так і фізичні компоненти, наприклад, через фальсифікацію сенсорних даних у робототехнічних системах [9, 10]. Таким чином, дослідження фокусуються на розробці стратегій протидії, які враховують специфіку реального часу, аби мінімізувати ризики без значного зниження продуктивності.

Сучасні механізми захисту безпеки в КФС РЧ, розроблені в рамках міждисциплінарних досліджень, базуються на ідеї моніторингу аномальної поведінки, оскільки експлуатація вразливостей генерує відхилення від нормального функціонування програми. Запропонувано низку примітивів безпеки під час виконання, спрямованих на протидію атакам з пошкодженням пам'яті [11, 12]. Ці примітиви адаптовані для виявлення конкретних типів загроз. Цілісність потоку даних запобігає несанкціонованій модифікації даних. Цілісність керуючого потоку захищає від перехоплення контролю виконання. Цілісність вказівників протидіє атакам на маніпуляцію адресами пам'яті. У дослідницькому фокусі ці механізми реалізуються шляхом інтеграції додаткових інструкцій перевірки в бінарний код програми на етапі компіляції, що дозволяє динамічно

оцінювати відповідність поведінки системи заздалегідь визначеній політиці безпеки [13, 14]. Під час виконання програми ці інструкції моніторять ключові операції, виявляючи відхилення, які сигналізують про потенційну атаку. Емпіричні оцінки показують, що впровадження таких примітивів значно посилює стійкість КФС РЧ до кіберзагроз, особливо в критичних застосуваннях, де традиційні захисні заходи з універсальних систем виявляються неефективними через високі додаткові витрати [15, 16]. Таким чином, зусилля дослідників спрямовані на оптимізацію цих механізмів для реального часу, забезпечуючи баланс між безпекою, ресурсоефективністю та дотриманням дедлайнів, що відкриває перспективи для подальших інновацій у сфері захищених кіберфізичних технологій [17, 18].

Таким чином, при проєктуванні операційних систем реального часу в кіберфізичних системах на основі компонентного підходу важливими завданнями виступають завдання з планування реального часу, забезпечення зменшення складності процесу, покращенні управління пам'яттю апаратно-прискорених пристроїв та спільне проєктування керування і планування.

Формулювання цілей статті

Метою роботи є підвищення рівня безпеки кіберфізичних систем реального часу за рахунок розширення та адаптації механізмів захисту цілісності потоків даних, керуючого потоку та вказівників, що мінімізує вплив на продуктивність та забезпечує дотримання часових вимог.

Виклад основного матеріалу

Забезпечення високого рівня захищеності кіберфізичних систем реального часу є однією з ключових проблем сучасних вбудованих обчислювальних систем, оскільки вимоги до інформаційної безпеки безпосередньо накладаються на жорсткі часові обмеження виконання завдань. На відміну від загального призначення обчислювальних платформ, у КФС порушення часових гарантій може призводити не лише до логічних помилок, а й до фізичних наслідків, що істотно підвищує ціну будь-яких додаткових обчислювальних витрат. У цьому контексті застосування традиційних механізмів безпеки без урахування специфіки планування завдань реального часу є методологічно необґрунтованим і не дозволяє забезпечити формальні гарантії своєчасності.

Переважна більшість існуючих підходів до захисту програмного забезпечення орієнтована на максимізацію рівня безпеки за умови фіксованого або емпірично допустимого зниження продуктивності. Такий підхід фактично ігнорує той факт, що в системах реального часу продуктивність не є оптимізованою величиною, а виступає обмеженням, яке не може бути порушене. У результаті проблема інтеграції механізмів безпеки зводиться до пошуку допустимого компромісу, тоді як задача оптимального розподілу обчислювальних ресурсів між функціональними та захисними операціями формально не ставиться і, відповідно, не розв'язується. Разом з тим фундаментальною особливістю систем реального часу є використання консервативних оцінок найгіршого часу виконання завдань, які необхідні для забезпечення часової коректності за будь-яких допустимих умов виконання. Така консервативність неминуче призводить до появи резервного часу виконання, який є побічним, але систематичним результатом застосування статичних методів аналізу часу виконання. Однак за відсутності формального апарату цей резерв часу залишається неструктурованим і не може бути використаний як об'єкт оптимізації. Таким чином, для переходу від констатації наявності slack time до його цілеспрямованого використання в інтересах підвищення захищеності системи необхідно побудувати моделі, які дозволяють кількісно описати процес його формування, еволюції та споживання під час виконання завдань. Без такого опису будь-які рішення щодо включення або адаптації механізмів безпеки залишаються евристичними, а їх вплив на часові характеристики буде непередбачуваним. Відсутність формалізованих моделей унеможливлює постановку задачі оптимізації в строгому математичному сенсі. Особливу складність становить той факт, що ключові примітиви забезпечення цілісності виконання, які включають контроль потоку даних, керуючого потоку та вказівників, мають різну структурну природу. Потік даних описується інформаційними залежностями між об'єктами програми, керуючий потік - графом переходів між базовими блоками, а цілісність вказівників - відношеннями доступу до адресного простору. Кожен із цих примітивів генерує власний тип додаткових витрат і по-різному взаємодіє з моделями планування завдань. Відтак оптимізація можлива лише за умови їх формалізації у вигляді узгоджених структурних моделей, здатних відображати як функціональні, так і часові характеристики. Наявність slack time не гарантує можливості його безпечного використання без порушення дедлайнів. Для цього необхідна точна, але водночас консервативна оцінка доступного резерву часу під час виконання, що потребує врахування варіативності вхідних даних, умов виконання та внутрішньої структури програм. Саме тому побудова моделей у цій роботі спирається на поєднання статичних методів аналізу, зокрема символічного виконання, з динамічними методами профілювання, що дозволяє отримати верхні межі часових характеристик без втрати коректності.

Тому, оптимізація продуктивності та захищеності кіберфізичних систем не може бути досягнута шляхом ізольованого вдосконалення окремих механізмів безпеки. Вона потребує побудови цілісного набору математичних і графових моделей, які формалізують взаємодію між завданнями реального часу, доступними обчислювальними ресурсами та структурою захисних примітивів. Лише на основі таких моделей стає можливим формальне формулювання оптимізаційної задачі балансування між рівнем захисту та гарантованою своєчасністю виконання, що і становить основну мету подальшого дослідження в межах даного параграфа. На рис. 1 зображено архітектуру узагальненої КФС з урахуванням особливостей оптимізації продуктивності та захищеності.

Важливою особливістю таких систем є використання консервативних оцінок часу виконання, які навмисно перевищують фактичні витрати обчислювального часу. Це призводить до появи резервного часу, який у більшості реалізацій не використовується. Розглядатимемо slack time як керований ресурс, який може бути використаний для виконання додаткових перевірок безпеки без порушення часових обмежень. Розробимо моделі механізмів цілісності потоків даних, керуючого потоку, вказівників і модель оптимізації продуктивності та захищеності КФС. Для моделі завдань КФС розглядатимемо множину завдань реального часу так:

$$Z_{CPS} = \left\{z_{CPS,1}, z_{CPS,2}, \dots, z_{CPS,N_{Z_{CPS}}}\right\}, \quad (1)$$

де $z_{CPS,i}$ - i – те завдання реального часу; $i = 1,2, \dots, N_{Z_{CPS}}$; $N_{Z_{CPS}}$ – кількість завдань КФС.

Рис.1. Узагальнена архітектура кіберфізичної системи з локалізацією області дослідження та механізмів оптимізації продуктивності й захищеності

Кожне завдання з множини завдань Z_{CPS} характеризується такими параметрами:

1) $c_i(t)$ - фактичний час виконання на момент часу t;

2) $O_{WCET,i}$ - консервативна оцінка найгіршого часу виконання;

3) D_i - відносний дедлайн;

4) T_i - період або мінімальний інтервал активації.

Резервний час визначимо так:

$$s_i(t) = O_{WCET,i} - c_i(t), \quad (2)$$

де $c_i(t)$ - фактичний час виконання на момент часу t; $i = 1,2, \dots, N_{Z_{CPS}}$; $N_{Z_{CPS}}$ – кількість завдань КФС; $O_{WCET,i}$ - консервативна оцінка найгіршого часу виконання.

Цей показник $s_i(t)$ згідно формули (2) використовується як динамічне обмеження для виконання додаткових захисних операцій.

Цілісність потоку даних (ЦПД, Data-Flow Integrity, DFI) спрямована на запобігання несанкціонованим модифікаціям або використанню даних у програмі. Модель механізму цілісності потоку даних задамо так:

$$G_{d,j} = \left(V_{d,j}, E_{d,j}\right), \quad (3)$$

де j – номер потоку даних; $V_{d,j}$ - множина змінних, буферів та регістрів; $V_{d,j} = \left\{v_{d,j,1}, v_{d,j,2}, \dots, v_{d,j,N_{V_{d,j}}}\right\}$; $k = 1,2, \dots, N_{V_{d,j}}$; $N_{V_{d,j}}$ - кількість елементів множини $V_{d,j}$; $E_{d,j}$ - допустимі інформаційні залежності «запис–читання»; $E_{d,j} \subset V_{d,j} \times V_{d,j}$.

Ребро $\left(v_{d,j,s}, v_{d,j,m}\right) \epsilon E_{d,j}$ $(s \leq k; m \leq k)$, що означає дозвіл значенню записаному у $v_{d,j,s}$ використати для формування значення $v_{d,j,m}$.

Умова цілісності полягає в тому, що кожне зчитування даних має відповідати дозволеному попередньому запису. Обчислювальні витрати перевірок ЦПД для завдання оцінимо так:

$$O_{\text{ЦПД},i} = k_d \cdot \left|E_{d,i}^1\right|, \quad (4)$$

де k_d - середня вартість перевірки однієї залежності; $E_{d,j}$ - допустимі середні значення залежності «запис–читання».

Для врахування часових обмежень введемо бінарну змінну активації:

$$A_{\text{ЦПД},i} = \begin{cases} 1, \text{якщо } O_{\text{ЦПД},i} \le s_i(t); \\ 0, \text{інакше}, \end{cases} \quad (5)$$

де $O_{\text{ЦПД},i}$ – обчислювальні витрати перевірок ЦПД для завдання; $s_i(t)$ – резервний час.

Таким чином, перевірки виконуються лише за наявності достатнього резерву часу.

Розглянемо приклад графа потоку даних в завданні керування. Нехай дано чотири елементи множини $V_{d,j}$ такі:

1) $v_{d,j,1}$ – сенсор;
2) $v_{d,j,2}$ – фільтр;
3) $v_{d,j,3}$ – стан;
4) $v_{d,j,4}$ – актуатор.

А також, дано допустимі залежності так: $E_{d,j} = \{(v_{d,j,1}, v_{d,j,2}), (v_{d,j,2}, v_{d,j,3}), (v_{d,j,3}, v_{d,j,4})\}$. Тоді, графічно отримуємо послідовність $v_{d,j,1} \to v_{d,j,2} \to v_{d,j,3} \to v_{d,j,4}$, яка відображає граф потоку даних. Цей граф описується матрицею суміжності A_j так:

$$A_j = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad (6)$$

де рядок – це джерело запису; стовпець – це ціль читання; ненульовий елемент означає дозволений потік даних.

Матриця інцидентності повинна враховувати, граф є орієнтований, тому значення її елементів будуть визначатись так:

$$B_j = |b_{j,s,l}|, b_{j,s,l} = \begin{cases} -1, \text{якщо } v_{d,j,s} \text{ є початковою вершиною дуги } e_{d,j,l}; \\ +1, \text{якщо } v_{d,j,s} v_{d,j,s} \text{ є кінцевою вершиною дуги } e_{d,j,l}; \\ 0, \text{інакше}, \end{cases} \quad (7)$$

де j – номер потоку даних; $V_{d,j}$ - множина змінних, буферів та регістрів; $V_{d,j} = \left\{v_{d,j,1}, v_{d,j,2}, \dots, v_{d,j,N_{V_{d,j}}}\right\}$; $k = 1,2, \dots, N_{V_{d,j}}$; $N_{V_{d,j}}$ - кількість елементів множини $V_{d,j}$; $E_{d,j}$ - допустимі інформаційні залежності «запис–читання»; $E_{d,j} \subset V_{d,j} \times V_{d,j}$; $E_{d,j} = \left\{e_{d,j,1}, e_{d,j,2}, \dots, e_{d,j,N_{E_{d,j}}}\right\}$; $u = 1,2, \dots, N_{E_{d,j}}$; $N_{E_{d,j}}$ - кількість елементів множини $E_{d,j}$.

Тоді, для ребер $e_{d,j,1} = (v_{d,j,1}, v_{d,j,2}), e_{d,j,2} = (v_{d,j,2}, v_{d,j,3}), e_{d,j,3} = (v_{d,j,3}, v_{d,j,4})$ матриця інцидентності має такий вигляд:

$$B_j = \begin{pmatrix} -1 & 0 & 0 \\ +1 & -1 & 0 \\ 0 & +1 & -1 \\ 0 & 0 & +1 \end{pmatrix}, \quad (8)$$

де рядки – це вершини графа; стовпці – дуги графа.

Для виявлення непрямих залежностей використаємо матрицю досяжності:

$$R_j = A_j^1 \vee A_j^2 \vee \dots \vee A_j^{m-1}, \quad (9)$$

де j – номер потоку даних; A_j - матриця суміжності.

Розглянемо такий приклад:

$$R_j = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad (10)$$

де $R_{j,s,w} = 1$ – це означає, що дані з $v_{d,j,s}$ можуть впливати на дані у $v_{d,j,w}$.

Будь-яка фактична залежність, яка є відсутньою в R_j, є порушенням ЦПД.

Здійснимо перевірку цілісності даних у матричному вигляді так:

$$A_j^1(t) \le R_j, \quad (11)$$

де $A_j^1(t)$ – матриця фактичних залежностей під час виконання і нерівність виконується покомпонентно.

Умова невиконання нерівності, яку задано за формулою (11), така:

$$\exists\, (p, w): a_{p,w}^1(t) = 1 \wedge r_{p,w} = 0. \quad (12)$$

Розглянемо врахування slack time. Кожній перевірці дуги відповідає вартість k_d. Нехай $E_j^1 \subseteq E_j$ – множина перевіряючих дуг. Тоді, обмеження задамо так:

$$\left|E_j^1\right| \cdot k_d \le S_j(t), \quad (13)$$

де k_d – вартість.

Таким чином, матриця перевірок є розрідженою підматрицею A_j (формула (6)).

Потік даних завдання КФС моделюється орієнтованим графом, вершини якого відповідають об’єктам даних, а ребра – допустимим інформаційним залежностям. Для формального подання графа використовується матриця суміжності, а для аналізу непрямих впливів – матриця досяжності. Перевірка цілісності потоку даних зводиться до порівняння матриці фактичних залежностей під час виконання з еталонною матрицею досяжності,

побудованою на етапі аналізу програми. Обсяг перевірок обмежується доступним резервом часу виконання завдання.

Розглянемо модель механізму цілісності керуючого потоку. Цілісність керуючого потоку (ЦКП, Control-Flow Integrity, CFI) забезпечує виконання програми виключно відповідно до дозволеного графа переходів.

Керуючий потік моделюватимемо графом так:

$$G_{d,j} = (V_{c,j}, E_{c,j}), \tag{14}$$

де вузли відповідають базовим блокам програми, а ребра – допустимим переходам між ними; j – номер потоку даних; $V_{c,j}$ - множина базових блоків програми; $V_{c,j} = \left\{v_{c,j,1}, v_{c,j,2}, \dots, v_{c,j,N_{V_{c,j}}}\right\}$; $k = 1,2, \dots, N_{V_{c,j}}$; $N_{V_{c,j}}$ - кількість елементів множини $V_{c,j}$; $E_{c,j}$ – множина допустимих переходів між базовими блоками програми.

Перевірка полягає у валідації наступної адреси виконання:

$$A_{p,next} = F_A(A_{p,current}), \tag{15}$$

де $A_{p,next}$ – наступна адреса в керуючому потоці; $A_{p,current}$ – поточна адреса в керуючому потоці; F_A – функція перетворення адрес.

Для зменшення додаткових витрат вводиться дискретний рівень деталізації перевірок так:

$$L_{\text{ЦКП}} \in \{0,1,2\}, \tag{16}$$

де 0 - перевірки вимкнені; 1 - груба перевірка; 2 - детальна перевірка.

Витрати перевірок визначимо так:

$$O_{\text{ЦКП},j}(L) = k_c \cdot \left|E^1_{c,j}(L)\right|, \tag{17}$$

де k_c – коефіцієнт вартості.

Рівень $L_{\text{ЦКП}}$ обирається таким чином, щоб максимізувати захищеність за наявного slack time.

Модель механізму цілісності вказівників і пам'яті будемо розглядати в контексті спроможності здійснювати контроль коректності вказівників під час доступу до пам'яті.

Для кожного завдання $z_{CPS,i}$ з множини завдань реального часу Z_{CPS} (формула (1)) визначимо множину допустимих адрес:

$$A_j^{valid} \subseteq A, \tag{18}$$

де $z_{CPS,i}$ - i – те завдання реального часу; $i = 1,2, \dots, N_{Z_{CPS}}$; $N_{Z_{CPS}}$ – кількість завдань КФС.

Умову цілісності можна задати так:

$$ptr \in A_j^{valid}, \tag{19}$$

де ptr – адреса вказівника.

Загальні витрати перевірок оцінюватимемо так:

$$O_{ptr,j} = k_p \cdot N_{ptr,j}, \tag{20}$$

де k_p – коефіцієнт вартості; $N_{ptr,j}$ — кількість операцій розіменування вказівника j – того потоку.

З метою адаптації до часових обмежень застосовується вибіркова перевірка лише критичних доступів, що формалізується за допомогою бінарних змінних вибору.

Згідно розроблених попередніх трьох моделей цілісних потоків даних, керуючого потоку та цілісності вказівкиків розробимо узагальнену модель оптимізації балансування безпеки та продуктивності в КФС.

Для кожного завдання формуємо вектор параметрів безпеки так:

$$q_j = \begin{bmatrix} A_{\text{ЦПД},j} \\ L_{\text{ЦКП},j} \\ B_{ptr,j} \end{bmatrix}, \tag{21}$$

де $A_{\text{ЦПД},j}$ – інформація щодо цілісних потоків даних; $L_{\text{ЦКП},j}$ – інформація щодо керуючого потоку; $B_{ptr,j}$ – інформація щодо вказівників.

Рівень захищеності системи задамо функцією так:

$$S_1(Q) = \sum_{j=1}^{n} w_j \cdot F_2(q_j), \tag{22}$$

де w_j – коефіцієнти критичності завдань; F_2 – функція оцінювання захищеності потоків даних, керуючого потоку та вказівників; n – кількість потоків; j – номер потоку даних; Q – множина завдань, яка складається з елементів q_j; $j = 1,2, \dots, n$.

Задачу оптимізації балансування безпеки та продуктивності в КФС визначимо так:

$$\max_Q S_1(Q), \tag{23}$$

за умови

$$c_j(t) + O_{\text{ЦПД},j} + O_{\text{ЦКП},j}(L) + O_{ptr,j} \le O_{WCET,j} \tag{24}$$

та глобальних планувальних обмежень і при цьому $c_j(t)$ - фактичний час виконання на момент часу t; $j = 1,2, \dots, N_{Z_{CPS}}$; $N_{Z_{CPS}}$ – кількість завдань КФС; $O_{WCET,j}$ - консервативна оцінка найгіршого часу виконання.

Таким чином, розроблено математичні моделі завдань КФС, а також трьох ключових механізмів забезпечення цілісності виконання. Запропоновано узагальнену оптимізаційну модель, що дозволяє динамічно балансувати між рівнем захищеності та часовими обмеженнями шляхом використання резервного часу виконання. Отримані моделі створюють теоретичну основу для подальшої розробки методу оптимізації та його експериментальної перевірки.

Подамо метод оптимізації продуктивності та захищеності КФС з використанням покрокового підходу.

Крок 1. Моделювання системи та визначення обмежень.

1.1. На цьому кроці потрібно розробити формальну модель КФС РЧ, причому початковими вимогами є завдання реального часу з жорсткими дедлайнами, потоки даних, потоки керування та критичні ресурси.

1.2. Визначити WCET для кожного завдання.

1.3. Визначити slack time – час, що залишається після виконання завдань до дедлайнів.

1.4. Задокументувати критичні точки для забезпечення безпеки, зокрема: контрольні точки цілісності; важливі покажчики; контексти керування.

Метою цього кроку є формування чіткого розуміння структури задач, ресурсів і часових обмежень.

Крок 2. Аналіз потенційних механізмів захисту.

2.1. Виділити механізми перевірки цілісності потоків даних, потоку керування,

асинхронного керування, контекстуальних покажчиків.

2.2. Оцінити витрати WCET для кожного механізму безпеки.

2.3. Визначити, які механізми можуть бути адаптовані під наявний slack time.

Метою цього кроку є встановлення можливості щодо перенесення обчислювальних витрат без порушення термінів виконання завдань КФС.

Крок 3. Оцінка та використання Slack Time.

3.1. Виконати консервативну оцінку slack для всіх завдань, зокрема: статичний аналіз WCET і профілювання реальних даних; символічне виконання для точного визначення резервів часу.

3.2. Визначити політики динамічного використання slack для безпеки, зокрема: виконання додаткових перевірок, якщо є вільний час; пріоритетне перенесення витрат захисту у час, що не впливає на терміни виконання.

Метою третього кроку є досягнення максимально ефективного використання наявного часу для забезпечення захисту.

Крок 4. Адаптивне планування завдань.

4.1. Впровадити моделі планування завдань, що враховують консервативні WCET, динамічно доступний slack, апаратну підтримку динамічного перемикання коду.

4.2. Розробити політики адаптивного розподілу ресурсів для перевірок цілісності потоків даних, асинхронного керування, захисту контексту та покажчиків

Метою четвертого кроку є забезпечення балансу між продуктивністю та безпекою через динамічне управління ресурсами.

Крок 5. Інтеграція просторової та часової безпеки.

5.1. Розмежувати ресурси для забезпечення просторової ізоляції для захисту критичних структур даних та контролю доступу до покажчиків і контекстів.

5.2. Застосувати часову ізоляцію для виконання захисних перевірок у межах slack time, мінімізації впливу механізмів безпеки на основний граф завдань.

Метою кроку є забезпечення гарантій того, що безпека не порушує встановлені терміни виконання завдань.

Крок 6. Динамічне управління безпекою.

6.1. Використання динамічного перемикання коду та контекстно-залежні механізми захисту для перевірки місць необхідного виконання і забезпечення контекстної чутливості покажчиків і потоків.

6.2. Адаптувати рівень захисту під наявний slack і поточне навантаження системи.

Метою кроку є забезпечення максимальної ефективності без компромісу з гарантованим виконанням.

Крок 7. Тестування та валідація.

7.1. Профілювання системи під реальними сценаріями.

7.2. Симуляція пікових навантажень і перевірка термінів виконання завдань та спрацювань механізмів безпеки у відведений slack.

7.3. Валідація балансу продуктивності та безпеки із застосуванням кількісних метрик, зокрема: завантаження процесора; виконання перевірок; використання часу slack.

Метою кроку є доведення ефективності запропонованого методу.

Ключові результати щодо кроків розробленого методу твкі: мінімізація витрат на перевірки цілісності потоків даних при гарантованому WCET; забезпечення асинхронного керування в реальному часі; цілісний захист потоку керування та контекстна чутливість; захист покажчиків із урахуванням планування та доступного slack.

Перший етап методу передбачає моделювання системи та визначення обмежень. На цьому етапі формується формальна модель КФС, яка включає всі завдання реального часу з жорсткими дедлайнами, критичні потоки даних та керування, а також апаратні ресурси, доступні для виконання цих завдань. Для кожного завдання виконується консервативна оцінка WCET, що гарантує виконання завдання у найгірших умовах без перевищення дедлайнів. Одночасно аналізується slack time – резерв часу, що залишається після виконання завдань до встановленого дедлайну. Для більш точного визначення slack проводиться профілювання системи, використовується символічне виконання та статистичний аналіз вхідних даних. На цьому етапі також виділяються критичні точки безпеки, де необхідно забезпечити цілісність даних, потоку керування та захист покажчиків.

Другий етап полягає у аналізі механізмів безпеки, які можуть застосовуватись для захисту КФС. До таких механізмів належать: перевірка цілісності потоків даних, захист потоку керування, перевірка асинхронного керування у реальному часі та захист покажчиків із урахуванням контексту. На цьому етапі

оцінюються витрати WCET для кожного механізму безпеки, щоб визначити, які перевірки можна виконати в рамках доступного slack. Це дозволяє не перевантажувати систему додатковими обчислювальними операціями, що могли б призвести до порушення дедлайнів.

Третій етап включає оцінку та використання slack time. Основною ідеєю цього етапу є перенесення обчислювальних витрат механізмів безпеки у вільний резерв часу. Використання методів символічного виконання дозволяє передбачити, які частини коду споживатимуть ресурси у пікові моменти, а профілювання вхідних даних дозволяє врахувати реальне навантаження системи. На основі цих даних формуються динамічні політики безпеки, які визначають, коли і які перевірки виконувати. Наприклад, перевірка цілісності покажчиків може виконуватися лише у моменти, коли завдання менш критичні щодо часу, або під час простоїв процесора, що не впливає на дедлайни критичних завдань.

Четвертий етап передбачає адаптивне планування завдань із урахуванням WCET і наявного slack. Завдання розподіляються динамічно на основі пріоритетів та критичності виконання. Апаратно підтримуване динамічне перемикання коду дозволяє виконувати захисні перевірки у різних контекстах без впливу на критичні завдання, а алгоритми динамічного розподілу ресурсів забезпечують перенесення обчислювальних витрат із пікових інтервалів у моменти доступного slack. Наприклад, перевірки цілісності потоку керування можуть бути виконані в перервах між виконанням завдань високого пріоритету, забезпечуючи цілісність системи без порушення дедлайнів.

П’ятий етап присвячений інтеграції просторової та часової безпеки. Просторова ізоляція передбачає відокремлення критичних структур даних та покажчиків від менш важливих частин системи, що дозволяє уникнути несанкціонованого доступу або пошкодження даних. Часова ізоляція забезпечує, що перевірки безпеки виконуються виключно у межах slack, що гарантує відсутність негативного впливу на основний граф завдань. Такий підхід дозволяє поєднати сувору безпеку із високою продуктивністю.

На шостому етапі здійснюється динамічне управління безпекою. Система адаптує рівень перевірок відповідно до поточного стану, навантаження та доступного slack. Контекстно-залежні перевірки дозволяють виконувати механізми безпеки тільки там, де це реально потрібно, наприклад, перевірка покажчиків і потоків даних активується лише при зміні критичного контексту. Це дозволяє мінімізувати витрати ресурсів на безпеку та уникнути перевантаження процесора.

Сьомий етап це тестування та валідація. На цьому етапі система проходить профілювання та стрес-тести, що дозволяє перевірити дотримання часових обмежень і ефективність застосованих механізмів безпеки. Виконується аналіз завантаження процесора, часу виконання перевірок та їх впливу на продуктивність. На основі отриманих даних коригуються динамічні політики безпеки та планування завдань для забезпечення оптимального балансу між продуктивністю та захищеністю.

Таким чином, запропонований метод дозволяє досягти рівня захисту КФС без порушення термінів виконання завдань, використовуючи доступні резерви часу, консервативне планування WCET, адаптивне планування завдань та динамічне управління механізмами безпеки. Метод забезпечує мінімізацію витрат на перевірки цілісності потоків даних, цілісний захист потоку керування, контекстну чутливість та захист покажчиків із урахуванням планування і динамічного розподілу ресурсів.

Схематично метод відображає логіку послідовного виконання етапів, тобто від моделювання та визначення обмежень до тестування та валідації, і демонструє, як кожен етап взаємодіє з іншими для забезпечення ефективного балансу між продуктивністю та безпекою. Такий підхід робить метод практичним для реальних КФС та забезпечує можливість його впровадження на сучасних апаратно-програмних платформах.

Використання даного методу в практичних системах дозволяє значно підвищити ефективність використання ресурсів, знизити енергоспоживання та забезпечити гарантовану своєчасність виконання критичних завдань, одночасно підтримуючи високий рівень захисту від потенційних атак на потоки даних, покажчики та керуючі контексти.

Розроблений метод оптимізації продуктивності та захищеності кіберфізичних систем на основі балансування завдань і ресурсів демонструє ефективний підхід до поєднання високого рівня безпеки з гарантованою своєчасністю виконання завдань у системах реального часу. Використання консервативних оцінок WCET, точного визначення slack time, адаптивного перенесення витрат механізмів безпеки у доступний резерв часу та динамічного управління контекстно-залежними перевірками дозволяє оптимізувати використання ресурсів, мінімізувати додаткові обчислювальні витрати та забезпечити цілісність потоків даних, захист потоку керування і покажчиків.

Метод має комплексний характер і включає моделювання системи, аналіз механізмів безпеки, адаптивне планування завдань, інтеграцію просторової та часової ізоляції, а також тестування та валідацію, що забезпечує практичну застосовність для реальних КФС. Застосування цього підходу дозволяє не тільки підвищити надійність і стійкість системи до атак та збоїв, але й забезпечує ефективне використання ресурсів і оптимізацію продуктивності.

Таким чином, запропонований метод є ефективним інструментом для проєктування захищених та продуктивних кіберфізичних систем, здатних працювати у жорстких умовах реального часу, і може бути безпосередньо використаний як методична основа для подальших досліджень та практичної реалізації в різних галузях промисловості та інженерії.

На відміну від традиційних методів захисту та оптимізації кіберфізичних систем, які здебільшого застосовують жорстке резервування ресурсів або постійні механізми контролю без урахування часу виконання,

запропонований метод використовує динамічне балансування завдань і ресурсів із врахуванням конкретних часових резервів (slack time). Це дозволяє переносити обчислювальні витрати перевірок цілісності потоків даних, керуючого потоку та вказівників у доступні часові резерви без порушення жорстких часових обмежень.

Класичні підходи, як правило, не інтегрують профілювання вхідних даних, символічне виконання та апаратно підтримане динамічне перемикання коду, що обмежує їхню ефективність та гнучкість у реальному часі. Запропонований метод забезпечує просторову й часову узгодженість безпеки та адаптивні політики захисту, що дає змогу досягти оптимального співвідношення між високим рівнем захисту та гарантованою своєчасністю виконання, тобто результат, недоступний для класичних методів.

Експерименти

Для оцінювання ефективності методу було змодельовано такі сценарії.

Сценарій 1. Низьке навантаження. Невелика кількість задач, значний slack time. Очікується активація повного набору механізмів безпеки.

Сценарій 2. Середнє навантаження. Slack time обмежений. Активуються лише перевірки цілісності даних.

Сценарій 3. Високе навантаження. Slack time близький до нуля. Механізми безпеки мінімізуються для збереження дедлайнів.

Результати експериментів подані в табл. 1 та табл. 2.

Таблиця 1

Вплив slack time на рівень безпеки

Сценарій	Середній slack time, мкс	Активні механізми безпеки	Пропущені задані терміни виконання
1	4200	ЦПД + ЦКП + вказівник	0
2	1800	ЦПД	0
3	200	Мінімальні	0

Таблиця 2

Додаткові витрати механізмів безпеки

Рівень безпеки	Додатковий час, % від WCET
Низький	1–2 %
Середній	4–6 %
Високий	8–12 %

На графіку з рис. 2 спостерігається зменшення відносних додаткових витрат безпеки при збільшенні slack time. Це пояснюється тим, що наявність резервного часу дозволяє виконувати додаткові перевірки цілісності потоків даних, керуючого потоку та вказівників без перевищення WCET завдань. При низькому slack time додаткові витрати максимально високі, оскільки більшість перевірок виконуються синхронно із основним потоком завдань, тоді як при середньому та високому slack time частина перевірок переноситься на вільні інтервали, що зменшує відносне навантаження на виконання критичних завдань.

Ступінчаста залежність на рис. 3 демонструє, що при малому резерві часу рівень безпеки низький через обмеження на виконання додаткових перевірок. При збільшенні slack time можливе підвищення рівня безпеки до середнього та високого, оскільки з'являється можливість виконати більше перевірок цілісності без порушення встановлених термінів виконання. Таким чином, графік ілюструє ефективність використання наявного резервного часу для оптимального балансування між безпекою та продуктивністю системи.

Графіки на рис. 2 і рис. 3 демонструють ключові взаємозв'язки між доступним slack time та реалізацією механізмів безпеки в кіберфізичних системах реального часу. Графік на рис. 2 показує, що додаткові витрати на безпеку зростають зі зменшенням slack time, при цьому використання резервного часу дозволяє мінімізувати додаткове навантаження на систему. Графік на рис. 3 ілюструє адаптивний рівень безпеки. При малому slack time система підтримує базовий захист, у середньому резерві можна застосувати помірний рівень перевірок, а при великому slack time досягається максимальний рівень безпеки без порушення дедлайнів. Разом ці графіки підтверджують ефективність підходу до балансування продуктивності та безпеки через динамічне використання доступного часу.

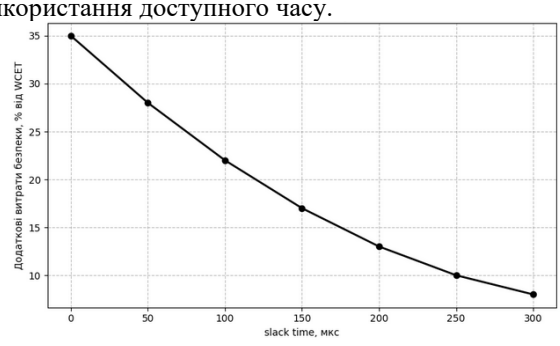

Рисунок 2 – Залежність додаткових витрат безпеки від slack time

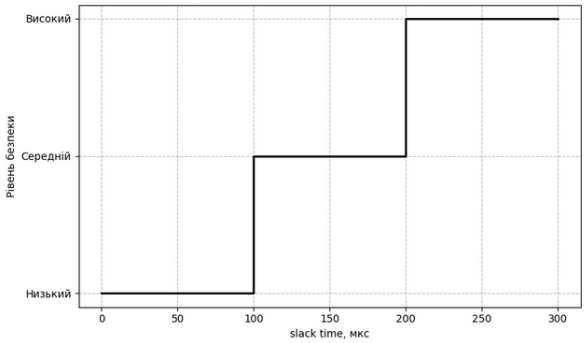

Рисунок 3 - Рівні безпеки залежно від slack time

Отримані результати підтверджують, що запропонований метод дозволяє ефективно використовувати slack time для адаптивної активації механізмів безпеки без порушення жорстких часових обмежень. Реалізований прототип демонструє практичний баланс між продуктивністю та захищеністю КФС реального часу, що підтверджує доцільність застосування методу в критичних вбудованих системах.

Проведені експериментальні дослідження переконливо підтвердили ефективність запропонованого підходу до оптимізації продуктивності та захищеності кіберфізичних систем реального часу. У межах усіх розглянутих сценаріїв виконання завдань жорсткі часові обмеження було повністю дотримано, що свідчить про коректність роботи планувальника реального часу та адекватність консервативної оцінки WCET. Водночас рівень безпеки системи динамічно адаптувався до величини доступного slack time, забезпечуючи активацію додаткових механізмів захисту за наявності часових резервів і їх мінімізацію за умов високого навантаження. Таким чином, реалізований прототип КФС РЧ на практиці демонструє доцільність і життєздатність розробленого методу, а також підтверджує можливість досягнення збалансованого поєднання високого рівня захищеності та гарантованої своєчасності виконання завдань у критичних КФС РЧ.

Висновки з даного дослідження
і перспективи подальших розвідок у даному напрямі

Проведений аналіз існуючих підходів до забезпечення безпеки кіберфізичних систем реального часу показав, що традиційні методи, засновані на жорсткому резервуванні ресурсів або постійному застосуванні механізмів контролю, призводять до значних додаткових часових витрат і знижують ефективність використання ресурсів системи. Розроблено метод використання консервативних оцінок WCET і slack time для адаптивного перенесення накладних витрат механізмів безпеки у доступні часові резерви, що дозволяє забезпечити необхідний рівень захисту без порушення жорстких часових обмежень виконання завдань. Запропоновано адаптивні механізми захисту цілісності потоків даних, потоку керування та покажчиків, інтегровані з плануванням завдань і динамічним розподілом ресурсів, які забезпечують контекстну чутливість перевірок і мінімізацію додаткових обчислювальних витрат.

Розроблено комплексну модель та архітектуру кіберфізичної системи з інтеграцією просторової й часової ізоляції та динамічного управління рівнями безпеки, що підтверджує практичну реалізованість і масштабованість запропонованого методу на сучасних апаратно-програмних платформах. За результатами експериментальних досліджень встановлено, що запропонований метод забезпечує гарантовану своєчасність виконання критичних завдань, зменшення накладних витрат безпеки та підвищення ефективності використання обчислювальних і енергетичних ресурсів при збереженні високого рівня захищеності кіберфізичної системи.

Подальші дослідження можуть бути спрямовані на інтеграцію запропонованого методу з апаратно-прискореними платформами, такими як GPU, FPGA та AI-акселератори, для підвищення ефективності балансування обчислювальних витрат і накладних витрат механізмів безпеки. Також актуальним є розвиток адаптивних алгоритмів планування завдань із прогнозуванням поведінки системи під час пікових навантажень, що дозволить підвищити стійкість КФС до непередбачуваних ситуацій і атак у режимі реального часу. Перспективним є застосування методів машинного навчання для контекстного визначення рівня безпеки та динамічного налаштування механізмів захисту без втрати продуктивності. Додатково, важливим напрямом є розширення моделі системи для підтримки кооперативних і розподілених КФС, де синхронізація та захист між вузлами суттєво впливають на загальну ефективність і безпеку.

Література

1. Kozelskiy O., Kashtalian A., Stetsyuk V., Martiniuk D., Sachenko A. A model of an intelligent clustering system with an external module for the architecture of RTOS with intensive changes of states regarding their flexibility and balancing. *1st International Workshop on Advanced Applied Information Technologies: (AdvAIT-2024),* Khmelnytskyi, Ukraine and Zilina, Slovakia, December 5, 2024. Vol. 3899. P. 234-243. URL: https://ceur-ws.org/Vol-3899/paper21.pdf

2. Kozelskiy O., Drozd A., Savenko B., Gaj P. A model for probabilistic monitoring and proactive restart of real-time operating systems under intensive state changes in cyber-physical systems. *2nd International Workshop on Intelligent & CyberPhysical Systems: (ICyberPhyS 2025),* Khmelnytskyi, Ukraine, July 4, 2025. Vol. 4013. P. 198-210. URL: https://ceur-ws.org/Vol-4013/paper16.pdf

3. Savenko O., Gaj P., Sierhieiev Ye. Detection of buffer overflow vulnerabilities in system software based on a graph and transformer model. *AISSLE-2025: The International Workshop on Applied Intelligent Security Systems in Law Enforcement*, October, 30–31, 2025, Vinnytsia, Ukraine. Pp. 292-305. URL: https://ceur-ws.org/Vol-4126/paper17.pdf

4. Sierhieiev Y., Paiuk V., Savenko O., Drozd A. Improvement of effectiveness for Static Application Security Testing for detection of SQL Injection vulnerabilities. *IEEE 14th International Conference on Dependable Systems, Services and Technologies (DESSERT-2024)* : Proceedings. Athens, Greece, October 11–13, 2024. Pp. 1–6. DOI: https://doi.org/10.1109/DESSERT65323.2024.11122171

5. Yuce B., Schaumont P., Witteman M. Fault Attacks on Secure Embedded Software: Threats, Design, and Evaluation. J. Hardw. *Syst. Secur*. 2018. Vol. 2. Pp. 111–130.

6. Mishra J., Sahay S.K. Modern Hardware Security: A Review of Attacks and Countermeasures. *arXiv* ***2025****, arXiv*:2501.04394. http://arxiv.org/abs/2501.04394

7. Savenko B., Kashtalian A., Lysenko S., Savenko O. Malware Detection By Distributed Systems with Partial Centralization, *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing*

Systems: Technology and Applications (IDAACS), Dortmund, Germany, 2023. Pp. 265-270. DOI: 10.1109/IDAACS58523.2023.10348773

8. Shuvo A.M., Zhang T., Farahmandi F., Tehranipoor M. A comprehensive survey on non-invasive fault injection attacks. *Cryptol. ePrint Arch*. 2023.

9. Gangolli A., Mahmoud Q.H., Azim A. A systematic review of fault injection attacks on IOT systems. *Electronics.* 2022, 11, 2023.

10. Kazemi Z., Hely D., Fazeli M., Beroulle V. A Review on Evaluation and Configuration of Fault Injection Attack Instruments to Design Attack Resistant MCU-Based IoT Applications. *Electronics*. 2020, 9, 1153.

11. Barenghi A., Breveglieri L., Koren I., Naccache D. Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures. *Proc. IEEE.* 2012, 100, 3056–3076.

12. Canella C., Van Bulck J., Schwarz M., Lipp M., Von Berg B., Ortner P., Piessens F., Evtyushkin D., Gruss D. A systematic evaluation of transient execution attacks and defenses. *In Proceedings of the 28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA, USA, 14–16 August 2019. Pp. 249–266.

13. Bedratyuk L., Savenko O., *MATCH Commun. Math. Comput. Chem.* 2018, 79, 407–414.

14. Xiong W., Szefer J. Survey of transient execution attacks and their mitigations. *ACM Comput. Surv*. (CSUR) 2021, 54, 1–36.

15. Agoyan M., Dutertre J.M., Naccache D., Robisson B., Tria A. When Clocks Fail: On Critical Paths and Clock Faults. In Smart Card Research and Advanced Application, *Proceedings of the 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010,* Passau, Germany, 14–16 April 2010, Proceedings; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6035, Pp. 182–193.

16. Claudepierre L., Péneau P.Y., Hardy D., Rohou E. TRAITOR: A Low-Cost Evaluation Platform for Multifault Injection. *In Proceedings of the 2021 International Symposium on Advanced Security on Software and Systems, Virtual Event*, Hong Kong, 7 June 2021. Pp. 51–56.

17. Denysiuk D., Savenko O., Lysenko S., Savenko B., Kashtalian A. Method for Detecting Steganographic Changes in Images Using Machine Learning. *13th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Athens, Greece, 2023. Pp. 1-6. DOI: 10.1109/DESSERT61349.2023.10416453

18. Korak T., Hoefler M. On the Effects of Clock and Power Supply Tampering on Two Microcontroller Platforms. *In Proceedings of the 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, Busan*, Republic of Korea, 23 September 2014. Pp. 8–17.

References

1. Kozelskiy O., Kashtalian A., Stetsyuk V., Martiniuk D., Sachenko A. A model of an intelligent clustering system with an external module for the architecture of RTOS with intensive changes of states regarding their flexibility and balancing. *1st International Workshop on Advanced Applied Information Technologies: (AdvAIT-2024),* Khmelnytskyi, Ukraine and Zilina, Slovakia, December 5, 2024. Vol. 3899. P. 234-243. URL: https://ceur-ws.org/Vol-3899/paper21.pdf

2. Kozelskiy O., Drozd A., Savenko B., Gaj P. A model for probabilistic monitoring and proactive restart of real-time operating systems under intensive state changes in cyber-physical systems. *2nd International Workshop on Intelligent & CyberPhysical Systems: (ICyberPhyS 2025),* Khmelnytskyi, Ukraine, July 4, 2025. Vol. 4013. P. 198-210. URL: https://ceur-ws.org/Vol-4013/paper16.pdf

3. Savenko O., Gaj P., Sierhieiev Ye. Detection of buffer overflow vulnerabilities in system software based on a graph and transformer model. *AISSLE-2025: The International Workshop on Applied Intelligent Security Systems in Law Enforcement*, October, 30–31, 2025, Vinnytsia, Ukraine. Pp. 292-305. URL: https://ceur-ws.org/Vol-4126/paper17.pdf

4. Sierhieiev Y., Paiuk V., Savenko O., Drozd A. Improvement of effectiveness for Static Application Security Testing for detection of SQL Injection vulnerabilities. *IEEE 14th International Conference on Dependable Systems, Services and Technologies (DESSERT-2024)* : Proceedings. Athens, Greece, October 11–13, 2024. Pp. 1–6. DOI: https://doi.org/10.1109/DESSERT65323.2024.11122171

5. Yuce B., Schaumont P., Witteman M. Fault Attacks on Secure Embedded Software: Threats, Design, and Evaluation. J. Hardw. *Syst. Secur*. 2018. Vol. 2. Pp. 111–130.

6. Mishra J., Sahay S.K. Modern Hardware Security: A Review of Attacks and Countermeasures. *arXiv 2025, arXiv*:2501.04394. http://arxiv.org/abs/2501.04394

7. Savenko B., Kashtalian A., Lysenko S., Savenko O. Malware Detection By Distributed Systems with Partial Centralization, *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Dortmund, Germany, 2023. Pp. 265-270. DOI: 10.1109/IDAACS58523.2023.10348773

8. Shuvo A.M., Zhang T., Farahmandi F., Tehranipoor M. A comprehensive survey on non-invasive fault injection attacks. *Cryptol. ePrint Arch*. 2023.

9. Gangolli A., Mahmoud Q.H., Azim A. A systematic review of fault injection attacks on IOT systems. *Electronics.* 2022, 11, 2023.

10. Kazemi Z., Hely D., Fazeli M., Beroulle V. A Review on Evaluation and Configuration of Fault Injection Attack Instruments to Design Attack Resistant MCU-Based IoT Applications. *Electronics*. 2020, 9, 1153.

11. Barenghi A., Breveglieri L., Koren I., Naccache D. Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures. *Proc. IEEE.* 2012, 100, 3056–3076.

12. Canella C., Van Bulck J., Schwarz M., Lipp M., Von Berg B., Ortner P., Piessens F., Evtyushkin D., Gruss D. A systematic evaluation of transient execution attacks and defenses. *In Proceedings of the 28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA, USA, 14–16 August 2019. Pp. 249–266.

13. Bedratyuk L., Savenko O., *MATCH Commun. Math. Comput. Chem.* 2018, 79, 407–414.

14. Xiong W., Szefer J. Survey of transient execution attacks and their mitigations. *ACM Comput. Surv*. (CSUR) 2021, 54, 1–36.

15. Agoyan M., Dutertre J.M., Naccache D., Robisson B., Tria A. When Clocks Fail: On Critical Paths and Clock Faults. In Smart Card Research and Advanced Application, *Proceedings of the 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010,* Passau, Germany, 14–16 April 2010, Proceedings; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6035, Pp. 182–193.

16. Claudepierre L., Péneau P.Y., Hardy D., Rohou E. TRAITOR: A Low-Cost Evaluation Platform for Multifault Injection. *In Proceedings of the 2021 International Symposium on Advanced Security on Software and Systems, Virtual Event*, Hong Kong, 7 June 2021. Pp. 51–56.

17. Denysiuk D., Savenko O., Lysenko S., Savenko B., Kashtalian A. Method for Detecting Steganographic Changes in Images Using Machine Learning. *13th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Athens, Greece, 2023. Pp. 1-6. DOI: 10.1109/DESSERT61349.2023.10416453

18. Korak T., Hoefler M. On the Effects of Clock and Power Supply Tampering on Two Microcontroller Platforms. *In Proceedings of the 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, Busan*, Republic of Korea, 23 September 2014. Pp. 8–17.