КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (27), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.27.710
УДК 004.42

Костюк Юлія Володимирівна
PhD in Computer Science
доцент кафедри інформаційної та кібернетичної
безпеки імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка, Київ, Україна
ORCID ID: 0000-0001-5423-0985
y.kostiuk@kubg.edu.ua

Довженко Надія Михайлівна
кандидат технічних наук, доцент,
доцент кафедри інформаційної та кібернетичної
безпеки імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка, Київ, Україна
ORCID ID:0000-0003-4164-0066
nadezhdadovzhenko@gmail.com

Мазур Наталія Петрівна
кандидат педагогічних наук, доцент,
доцент кафедри інформаційної та кібернетичної
безпеки імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка, Київ, Україна
ORCID ID: 0000-0001-7671-8287
n.mazur@kubg.edu.ua

Складанний Павло Миколайович
кандидат технічних наук, доцент
завідувач кафедри інформаційної та кібернетичної
безпеки імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка, Київ, Україна
ORCID ID: 0000-0002-7775-6039
p.skladannyi@kubg.edu.ua

Рзаєва Світлана Леонідівна
кандидат технічних наук, доцент,
доцент кафедри кафедри комп'ютерних наук
Київський столичний університет імені Бориса Грінченка, Київ, Україна
ORCID ID: 0000-0002-7589-2045
s.rzaieva@kubg.edu.ua

МЕТОДИКА ЗАХИСТУ GRID-СЕРЕДОВИЩА ВІД ШКІДЛИВОГО КОДУ ПІД ЧАС ВИКОНАННЯ ОБЧИСЛЮВАЛЬНИХ ЗАВДАНЬ

Анотація. Стаття присвячена захисту Grid-середовищ від шкідливого коду, який може бути інтегрований під час виконання обчислювальних завдань. Визначено основні проблеми, зокрема високі ризики зловмисного втручання, що можуть порушити роботу обчислювальних вузлів або паралізувати всю систему. Окремо розглянуто проблему поширення шкідливого коду, що може вивести вузли з ладу. Запропоновано методику захисту, що базується на автоматичній верифікації вихідного коду, яка дозволяє виявляти шкідливі програми до їх виконання без значного впливу на продуктивність. Розглянуто підходи до виявлення загроз, зокрема статичний і динамічний аналіз коду, а також використання алгоритмів машинного навчання для прогнозування атак. Інтеграція таких методик підвищить безпеку Grid-середовищ і сприятиме їхньому застосуванню в науці та промисловості. Модель управління пам'яттю описує комірки, що можуть бути в станах: вільна, виділена або помилкова, і враховує необхідність захисту пам'яті від несанкціонованого доступу. Перехід між станами здійснюється через визначені операції, що забезпечують безпеку даних. Архітектура захисту

включає безпечну бібліотеку, модуль динамічної верифікації та монітор звернень, що дає змогу ефективно моніторити і захищати систему від шкідливого програмного забезпечення. Стратегія безпеки базується на адаптивному управлінні політиками та статичній верифікації для запобігання загрозам, підвищуючи стійкість до кіберзагроз у реальному часі.

Ключові слова: Grid-середовище; шкідливий код; ризики зловмисного втручання; верифікація коду; статичний і динамічний аналіз; машинне навчання; безпека пам'яті; архітектура захисту; адаптивне управління політиками безпеки.

ВСТУП

Сучасний розвиток науки й технологій вимагає вирішення складних завдань, які потребують значних обчислювальних потужностей, недоступних в межах окремих обчислювальних центрів. Для забезпечення виконання таких завдань були створені гетерогенні середовища розподілених обчислень, що отримали назву Grid-середовища. Ці інфраструктури являють собою географічно розподілені системи, які об'єднують різнотипні ресурси, зокрема процесори, запам'ятовувальні пристрої, бази даних і комп'ютерні мережі. Їхньою особливістю є можливість надання користувачам доступу до цих ресурсів незалежно від їхнього фізичного розташування, що суттєво спрощує виконання масштабних обчислювальних завдань.

Grid-середовище часто порівнюють з електромережею, оскільки доступ до його ресурсів має бути таким же зручним і простим, як увімкнення пристрою в розетку. Це стало можливим завдяки інтеграції різних технологій і методів управління, які забезпечують ефективне використання розподілених обчислювальних потужностей [1] –[4], [6]. Однак активний розвиток і поширення Grid-технологій супроводжуються значними викликами в сфері інформаційної безпеки. Головною проблемою є можливість зловмисного втручання в роботу системи, яке може призвести до порушення функціонування великої кількості ресурсів і завдати шкоди багатьом користувачам. Особливу небезпеку становить поширення шкідливого коду у складі обчислювальних завдань. Такий код може вивести з ладу окремі обчислювальні вузли або навіть паралізувати всю систему в цілому. Крім того, Grid-середовище може бути використане як платформа для здійснення розподілених атак, що створює серйозну загрозу глобальній кібербезпеці.

У сучасних умовах проблема захисту Grid-середовищ набуває критичної актуальності. Для її вирішення необхідно розробляти ефективні методики, які враховують унікальні характеристики таких систем, зокрема їхню масштабність, різнорідність ресурсів і географічну розподіленість. Одним із перспективних напрямів є впровадження програмно-апаратних рішень, здатних забезпечувати постійний моніторинг і блокування шкідливого коду [5], [9]. Додатково слід застосовувати методи штучного інтелекту та машинного навчання для прогнозування потенційних загроз і виявлення аномалій у роботі системи.

Постановка проблеми. Захист Grid-середовищ є критично важливим, оскільки ці інфраструктури забезпечують вирішення складних завдань у науці, промисловості та інших сферах. Безпека таких систем має стратегічне значення для забезпечення надійності та безперервності їх роботи, а також для мінімізації ризиків зловмисного втручання. Наразі для аутентифікації користувачів і розробників обчислювальних завдань широко використовуються інфраструктури відкритих ключів [1], [7]. Проте ці підходи здатні лише фіксувати факт реалізації загроз, що може призвести до значних збитків у разі поширення шкідливого коду [3], [4].

Враховуючи обмеження традиційних методів безпеки, актуальним є застосування технічних засобів захисту, зокрема автоматичної верифікації вихідного коду, що мінімізує вплив на продуктивність системи та дозволяє оперативно виявляти шкідливий код до його виконання. Крім того, обчислювальні завдання для Grid-середовищ розробляються за допомогою різних мов програмування, таких як C, C#, Java та Python, при цьому мова C є найоптимальнішою для гетерогенних середовищ завдяки своїй універсальності, високій продуктивності та підтримці більшості апаратних платформ [8], [17], [18].

Таким чином, проблема полягає в розробці методики захисту Grid-середовищ від шкідливого коду в обчислювальних завданнях, яка ґрунтується на автоматичній верифікації вихідного коду, що дозволить значно підвищити рівень безпеки таких систем та знизити ризики їх використання в нових сферах. Сучасні дослідження зосереджені на інтеграції методів аналізу програмного забезпечення, зокрема статичного і динамічного аналізу, а також впровадженні алгоритмів машинного навчання для виявлення потенційних загроз. Тому необхідно розробити ефективну методику захисту, що відповідає міжнародним стандартам інформаційної безпеки та забезпечує надійність Grid-систем [5], [16].

Аналіз останніх досліджень і публікацій. Аналіз літературних джерел з теми захисту Grid-середовищ від шкідливого коду дозволяє виділити кілька основних напрямків у дослідженнях, зокрема проблему безпеки розподілених обчислювальних систем. Одним із провідних науковців, що досліджує цей напрямок, є Sharma et al. [1], які аналізують використання алгоритмів машинного навчання для прогнозування потенційних атак у Grid-середовищах, відзначаючи ефективність таких підходів у забезпеченні проактивного захисту. Вони підкреслюють важливість інтеграції технологій машинного навчання для виявлення шкідливих програм на етапі попереднього аналізу, що значно знижує ризики до моменту їх виконання.

Також цікаві роботи Chen et al. [2], що пропонують методи статичного та динамічного аналізу коду, орієнтуючись на виявлення вразливостей до виконання обчислювальних завдань. Вони відзначають, що комбінація статичного аналізу для перевірки вихідного коду і динамічного моніторингу під час виконання завдань дає змогу виявляти нові типи атак і швидко реагувати на них, що є важливим для захисту Grid-систем.

Gupta & Soni [3] досліджують стратегії управління пам’яттю в Grid-середовищах, зокрема, важливість захисту пам’яті від несанкціонованого доступу та витоків даних. Вони розробили модель управління пам’яттю, яка враховує захист від небажаних операцій, таких як читання конфіденційних даних, що є критично важливим для запобігання інформаційним витокам у розподілених системах.

У роботах Li & Yang [4] розглядається адаптивне управління політиками безпеки в реальному часі, що дозволяє на основі оцінки ризиків швидко коригувати параметри захисту, відповідно до змін умов або загроз. Вони наголошують на важливості інтеграції таких методик у реальні виробничі середовища для підвищення стійкості до кіберзагроз.

Крім того, у наукових працях Wang & Chen [5] та Ravichandran & Kumar [6] активно розглядаються архітектури захисту, що включають динамічні верифікаційні модулі та монітори звернень, які дозволяють захищати Grid-системи від шкідливого програмного забезпечення, зокрема, в умовах інтеграції різних платформ. Вони відзначають, що ці технології дозволяють зберегти високу продуктивність обчислювальних завдань при мінімальному ризику втручання.

Таким чином, сучасні дослідження зосереджені на використанні комбінованих методів, таких як автоматичне верифікування коду, аналіз пам’яті та впровадження

адаптивних політик безпеки, що дозволяє забезпечити більш ефективний захист Grid-систем від шкідливого коду, знижуючи ризики і збільшуючи стійкість до атак.

Мета статті. Метою статті є дослідження методів захисту Grid-середовищ від шкідливого коду, інтегрованого в процес обчислень. Особливу увагу приділено виявленню та запобіганню загрозам на етапах верифікації вихідного коду, моніторингу пам'яті та виявленню аномальних дій під час виконання обчислень. Зокрема, вивчаються ризики зловмисного втручання, що можуть паралізувати систему, та механізми поширення шкідливого коду між вузлами.

Завдяки статичному і динамічному аналізу коду, а також алгоритмам машинного навчання для прогнозування атак, підвищується точність виявлення загроз і скорочується час реакції. Створюється модель управління пам'яттю для захисту від несанкціонованого доступу та витоків даних через контроль операцій пам'яті. У рамках статті розглядається також роль пам'яті як важливого елемента захисту, зокрема, розробка моделей управління пам'яттю, які дозволяють здійснювати контроль за її станами та операціями, такими як виділення, звільнення, читання і запис, що допомагає уникнути витоків даних та зловмисних атак, включаючи відмову в обслуговуванні.

Запропонована архітектура безпеки включає безпечні бібліотеки, динамічну верифікацію та монітори звернень, що забезпечує сумісність з різними платформами та мінімальний вплив на продуктивність. Стратегії адаптивного управління політиками безпеки підвищують стійкість системи до кіберзагроз у реальному часі, що дозволяє розширити використання Grid-систем у науці та промисловості.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Сучасні Аналіз особливостей захисту Grid-середовищ від шкідливого коду, що виконується під час обчислювальних завдань, є важливим етапом у розробці методик захисту таких середовищ. Середовище розподілених обчислень (СРВ) має ряд специфічних характеристик, які впливають на застосування методів захисту від шкідливого коду. Перш за все, у таких середовищах використовуються численні обчислювальні вузли, що з одного боку забезпечує високу продуктивність, а з іншого — створює додаткові ризики для безпеки. Це обумовлює необхідність застосування спеціальних технічних засобів для захисту від шкідливого коду, які здатні ефективно взаємодіяти з різними платформами і забезпечувати мінімальний вплив на продуктивність системи [1], [5], [10].

Однією з ключових проблем є використання повільних каналів зв'язку між обчислювальними вузлами та диспетчером СРВ, що може сповільнити виявлення та нейтралізацію шкідливого коду. Крім того, Grid-середовище зазвичай призначене для виконання завдань, що вимагають великої кількості обчислень при малому обсязі вхідних даних та результату, який передається диспетчеру [2]. Це ставить додаткові вимоги до швидкодії методів захисту від шкідливого коду, адже необхідно забезпечити баланс між високою продуктивністю обчислень і ефективним захистом від загроз.

Середовище виконання обчислювальних завдань у Grid-середовищах є програмно та апаратно гетерогенним, що означає, що для забезпечення безпеки потрібно враховувати різні платформи та програмні інтерфейси операційних систем. Враховуючи це, необхідно розробляти методики, які забезпечать захист не лише на рівні окремих обчислювальних вузлів, а й на рівні всього середовища, не порушуючи його гнучкості та ефективності [3].

Виконувані в таких середовищах обчислювальні завдання повинні бути переносимими з однієї платформи на іншу, тобто не залежати від конкретного апаратного чи програмного забезпечення. У зв'язку з цим, розробка методик захисту повинна включати в себе створення універсальних верифікаторів вихідного коду, що дозволяють виявляти шкідливий код ще до його виконання на обчислювальних вузлах [4]. Важливо, що обчислювальні завдання у таких середовищах не повинні взаємодіяти з користувачами ЕОМ, на яких вони виконуються, що знижує ймовірність зовнішнього впливу на процес виконання завдання.

Захист від шкідливого коду в таких середовищах потребує інтеграції різних підходів, зокрема статичного і динамічного аналізу коду, а також застосування сучасних алгоритмів для виявлення потенційних загроз. Важливим є також врахування вимог до продуктивності системи, оскільки методи захисту не повинні істотно знижувати ефективність виконання обчислювальних завдань. Тому автоматична верифікація вихідного коду, що дозволяє мінімізувати ризики поширення шкідливого коду, є важливим елементом методики захисту Grid-середовищ [5].

Раніше розглядалися різні методи захисту Grid-середовища від шкідливого коду, що виконується під час обчислювальних завдань. Одним із таких методів є віртуалізація, за якої виконання програмного забезпечення (ПЗ) відбувається в спеціально створеному віртуальному середовищі, доступ до якого до реального апаратного забезпечення обмежений, що дозволяє створити певний рівень ізоляції від потенційно шкідливого коду. Іншим важливим методом є системи безпечного програмування, що передбачають розробку ПЗ за допомогою спеціальних мов програмування, котрі запобігають порушенню безпеки системи. Крім того, застосовується верифікація програмного забезпечення, яка включає перевірку його безпеки перед використанням, що дозволяє виявляти уразливості та попереджати можливі атаки. Підпис ПЗ, за якого автор підписує програмне забезпечення своїм електронним цифровим підписом (ЕЦП), дає змогу ідентифікувати автора і застосовувати відповідні нормативно-правові санкції у разі виявлення шкідливого коду. Важливою складовою захисту є також аналіз мережевої активності, що включає спостереження за мережевим трафіком учасників Grid-середовища, що дозволяє своєчасно виявляти та запобігати деяким типам мережевих атак [6].

Проте, за результатами досліджень, виявлено, що кожен з розглянутих методів має певні недоліки. Одним із основних є значне зниження продуктивності системи, що спостерігається, зокрема, при використанні повної емуляції, застосуванні систем безпечного програмування, трансляції в безпечне ПЗ або при використанні трасуючих моніторів [8]. Складність застосування таких методів також є важливою проблемою, оскільки верифікація, що вимагає участі людини, або використання коду, який містить доказ чи модель, потребує значних ресурсів та зусиль. Крім того, ці методи висувають підвищені вимоги до апаратного забезпечення та операційної системи, що ускладнює їх застосування в гетерогенних середовищах, оскільки віртуалізація на рівні додатків чи операційної системи, а також парвіртуалізація і часткова емуляція, потребують значних ресурсів та спеціального налаштування [9]. Більш того, ці методи не завжди забезпечують всебічний захист від шкідливого ПЗ, зокрема від переповнення буфера чи атак типу «відмова в обслуговуванні». Крім того, вони часто обмежують вибір засобів розробки, оскільки системи безпечного програмування та верифікація вихідного коду вимагають спеціалізованих інструментів і висококваліфікованих фахівців [10].

Багато з цих методів також не враховують специфіку Grid-середовища, що може призвести до зниження рівня безпеки або застосування надмірно жорсткої політики безпеки, що створює зайве навантаження на систему. У зв'язку з цим, одним із

перспективних підходів є впровадження автоматичної статичної верифікації на рівні вихідних кодів, оскільки цей метод не впливає на продуктивність ПЗ та дозволяє застосовувати програмне забезпечення в гетерогенних середовищах. Окрім того, для ефективного захисту від атак типу «відмова в обслуговуванні» необхідно використовувати додаткові засоби, що контролюють інтенсивність мережевого взаємодії, що дасть змогу забезпечити більш ефективний захист від мережевих загроз [11].

Методика захисту Grid-середовища від шкідливого коду під час виконання обчислювальних завдань передбачає комплексний підхід до забезпечення безпеки обчислювальних систем, що функціонують у межах розподіленого середовища. Основною метою цієї методики є виявлення та запобігання впливу шкідливих програм, які можуть порушити цілісність, конфіденційність або доступність оброблюваних даних. Першим етапом є побудова моделі порушника, що дозволяє оцінити потенційні загрози та виявити можливі вразливості в програмному забезпеченні [12]. Визначення та класифікація загроз сприяють формулюванню вимог до безпеки програмного забезпечення, яке виконується на обчислювальних вузлах Grid-системи. Важливим кроком є також створення формальної моделі коду, що дозволяє автоматизувати процес перевірки ПЗ на наявність уразливостей, здатних призвести до виконання довільного коду [13].

Методика захисту включає чітке визначення потенційних загроз, що можуть бути реалізовані через різні типи атак, зокрема атаки на цілісність інформації, що обробляється на обчислювальних вузлах, а також атаки, спрямовані на компрометацію обчислювальних завдань. Запобігання таким загрозам досягається через реалізацію низки заходів, зокрема через заборону використання інструкцій і програмних інтерфейсів, які можуть порушити цілісність компонентів системи, а також через верифікацію ПЗ для виявлення уразливостей, що можуть бути використані для виконання довільного коду. Застосування спеціалізованих програмних інтерфейсів для безпечного обміну даними з диспетчером Grid-середовища є важливим заходом для забезпечення захищеного комунікаційного каналу, який дозволяє мінімізувати ризики від несанкціонованого доступу до критичної інформації [6].

Для забезпечення захисту від атак на доступність методика включає обмеження на використання команд і програмних інтерфейсів, які можуть призвести до переведення обчислювальних вузлів у нештатний стан або до реалізації атак типу «відмова в обслуговуванні» (DoS) чи «розподілена відмова в обслуговуванні» (DDoS). Крім того, обмеження доступу до неініціалізованих областей оперативної пам’яті та заборона прямого доступу до даних на жорсткому диску чи інших носіях є критично важливими для запобігання зловмисним змінам оброблюваної інформації [7]. Для більш ефективного захисту також пропонується застосування сучасних методів нестандартного виявлення, включаючи використання алгоритмів машинного навчання та штучного інтелекту, що дозволяють виявляти підозрілі дії в реальному часі. Ці технології допомагають виявляти невидимі на перший погляд загрози, що можуть бути викликані змінами у поведінці програмного забезпечення або обчислювальних вузлів. Інтеграція таких систем дозволяє значно підвищити рівень захищеності та зменшити ризик ефективних атак на систему [10]. Загалом, методика захисту Grid-середовища від шкідливого коду передбачає багатоступеневий підхід, який включає як технічні, так і організаційні заходи, спрямовані на забезпечення безпеки обчислювальних завдань. Верифікація ПЗ та реалізація політик безпеки для обмеження доступу до системи є основними компонентами цієї методики, яка дозволяє створити надійне і стійке середовище для виконання розподілених обчислень в умовах сучасних кіберзагроз [6].

Методика захисту Grid-середовища від шкідливого коду під час виконання обчислювальних завдань передбачає комплексний підхід до забезпечення безпеки обчислювальних вузлів, що працюють в умовах гетерогенної середовища. Для ефективного захисту необхідно розробити та впровадити політику безпеки, яка враховує специфіку кожного вузла, зокрема апаратне та програмне забезпечення, операційну систему та особливості виконуваних завдань. Зокрема, програмне забезпечення повинно бути ретельно перевірено на відповідність вимогам безпеки, що забезпечує виключення можливості доступу до чутливих даних, а також порушення нормальної роботи системи в цілому. Крім того, важливим аспектом є регулярне оновлення компонентів програмного та апаратного забезпечення для забезпечення актуальності захисних механізмів. Враховуючи розподілену природу Grid-середовища, необхідно інтегрувати механізми виявлення та блокування шкідливого коду на рівні кожного вузла та через мережу. Це включає впровадження систем виявлення вторгнень (IDS), які здатні оперативно реагувати на підозрілі дії, а також використання криптографічних методів для захисту передачі даних між вузлами. Політика безпеки повинна включати елементи контролю доступу та аутентифікації, щоб мінімізувати ризики несанкціонованого втручання та доступу до важливих ресурсів [9]. Окрім традиційних методів захисту, таких як міжмережеві екрани та антивірусні програми, особливу увагу варто приділити впровадженню систем машинного навчання для прогнозування та виявлення нових, невідомих загроз. Інтеграція цих механізмів дозволить забезпечити високий рівень безпеки в Grid-середовищі та ефективно захистити обчислювальні ресурси від шкідливого коду [8].

Одним із ключових аспектів захисту є контроль доступу до пам’яті та даних, що обробляються, що дозволяє запобігти несанкціонованому доступу або модифікації інформації. Важливою частиною методики є верифікація вихідного коду на наявність уразливостей, які можуть призвести до виконання шкідливого коду. Це включає перевірку на відсутність прямого звернення до операційної системи, доступу до неініціалізованих ділянок пам’яті, а також наявності коду, здатного до самомодифікації. Необхідно приділяти увагу динамічному аналізу виконуваних програм для виявлення потенційно небезпечних операцій, які можуть бути пропущені під час статичної верифікації. Важливим елементом є застосування механізмів ізоляції обчислювальних середовищ, які обмежують вплив шкідливого коду на інші частини системи, знижуючи ймовірність його поширення. Для посилення захисту також застосовуються методи моніторингу та виявлення аномальних поведінкових патернів у реальному часі, що дозволяють своєчасно виявити підозрілі дії і запобігти потенційним загрозам. Інтеграція цих заходів забезпечує комплексний захист, який враховує різноманітні вектори атак і мінімізує ймовірність успішного виконання шкідливого коду в Grid-середовищах [11], [13] – [15].

Забезпечення захисту від шкідливого програмного забезпечення також вимагає ретельного моніторингу мережевого трафіку, що дозволяє вчасно виявити спроби витоку даних або несанкціонованого доступу. Крім того, важливо впроваджувати спеціалізовані засоби для контролю індексів при зверненні до масивів, перевірки діапазонів адрес пам’яті, а також контролю перетворень типів вказівників і викликів функцій операційної системи чи сторонніх бібліотек. Для підвищення ефективності захисту також використовуються механізми виявлення і запобігання вторгненням, які дозволяють в реальному часі ідентифікувати аномальні чи підозрілі дії, що можуть свідчити про спробу атаки. Застосування системи управління подіями безпеки дозволяє централізовано збирати й аналізувати дані про діяльність користувачів і програм, швидко реагуючи на можливі загрози. Важливо також регулярно оновлювати бази даних шкідливого програмного забезпечення та проводити аудит системи для виявлення нових

вразливостей, які можуть бути використані для атак. Інтеграція цих підходів дозволяє забезпечити багатоетапний захист, мінімізуючи ймовірність успішних атак і знижуючи ризики для цілісності та конфіденційності даних у Grid-середовищах [2], [3].

Методика захисту Grid-середовища від шкідливого коду під час виконання обчислювальних завдань передбачає застосування комплексного підходу, який охоплює як статичний, так і динамічний аналіз програмного забезпечення з метою виявлення потенційних загроз. Статичний аналіз полягає в перевірці вихідного коду для виявлення порушень безпекових політик, визначених для конкретного обчислювального завдання в середовищі Grid. Це включає перевірку абстрактної інтерпретації програми, при якій програма представлена у вигляді направленого графа, де вузли відповідають лінійним ділянкам коду, а дуги — гілкам потоку керування. У цьому контексті важливою складовою є уточнення діапазонів ймовірних значень змінних, що використовуються в програмі, залежно від вхідних даних та різних шляхів виконання коду [14]. Якщо на певному етапі виявляється, що значення змінних можуть порушити безпекові вимоги, ділянка коду позначається як потенційно небезпечна, і здійснюється подальша перевірка шляхів, що ведуть до цих ділянок.

Динамічний аналіз полягає у впровадженні моніторингу формальних властивостей програмного забезпечення в процесі його виконання. Такий моніторинг дозволяє виявляти спроби порушення політики безпеки в реальному часі. У разі виявлення порушень система моніторингу автоматично зупиняє виконання програми, тим самим запобігаючи можливим атакам або несанкціонованому доступу до даних [4], [5].

Особливу увагу в методиці захисту приділено забезпеченню надійної верифікації програмного забезпечення, яке використовує побітові операції, що є типовими для низькорівневих обчислень в Grid-середовищі. Для цього вводиться абстрактний домен двійкових n-вимірних векторів, який дозволяє здійснювати перевірку програм, що включають побітові зсуви, побітове виключаюче АБО, побітові операції І та НЕ. Завдяки використанню цього абстрактного домену стає можливою верифікація програмного забезпечення, що містить побітові операції [6], [16], [17]. Запропонований домен побудований як декартовий добуток n абстрактних доменів двійкових змінних і позначається як X^{An}, підхід дозволяє здійснювати точну перевірку програм, що виконують побітові операції, що є необхідним для забезпечення безпеки в Grid-середовищах під час виконання обчислювальних завдань:

$$X^{An} = \{x | x = \langle x_1, x_2, \dots, x_n \rangle, \text{де } x_i \in \{\bot, 0, 1, \top\}, i = 1, 2, \dots, n\}, \quad (1)$$

Під $\bot$ та $\top$ розуміються «мінімум» і «максимум» абстрактного домену двійкових змінних, тобто його найменший та найбільший елементи відповідно. Відношення порядку в X^{An} позначимо як $\leq$ та визначимо наступним чином:

$$\forall A \in X^{An}, \forall B \in X^{An}, A = \langle a_1, a_2, \dots, a_n \rangle, B = \langle b_1, b_2, \dots, b_n \rangle, \quad (2)$$

$$A \leq B \Leftrightarrow a_i \leq b_i \; \forall i \in \{1, 2, \dots, n\}, \quad (3)$$

$$\bot \leq 0 \leq \top, \bot \leq 1 \leq \top, \forall n \{\bot, 0, 1, \top\} \; n \leq n, \quad (4)$$

Операція абстракції f_a: $2^{V_n} \to X^{An}$ визначається наступним чином:

$$\forall C \in 2^{V_n}, A = f_a(C) \Leftrightarrow A = \langle a_1, a_2, \dots, a_n \rangle, \forall i \in \{1, 2, \dots, n\}, \quad (5)$$

$$a_i = \begin{cases} 0, \text{якщо } \forall c \in C, c = \langle c_1, c_2, \dots, c_n \rangle \; c_i = 0, \\ 1, \text{якщо } \forall c \in C, c = \langle c_1, c_2, \dots, c_n \rangle \; c_i = 1, \\ \top, \text{якщо } \exists b, c \in C, b = \langle b_1, b_2, \dots, b_n \rangle, c = \langle c_1, c_2, \dots, c_n \rangle \colon \; b_i \neq c_i, \\ \bot, \text{якщо } C = \emptyset. \end{cases} \quad (6)$$

Операція абстракції f_c: $X^{An} \to 2^{V_n}$ визначається наступним чином:

$$\forall A \in X^{An}, \quad A = \langle a_1, a_2, \dots, a_n \rangle, \forall c \in V_n, \quad c = \langle c_1, c_2, \dots, c_n \rangle, \quad (7)$$

ISSN 2663 - 4023

$$C \in f_c(A) \Leftrightarrow \forall i \in \{0,1,\dots,n\}\ C_i \in \begin{cases} \{0\}, \text{якщо } a_i = 0, \\ \{1\}, \text{якщо } a_i = 1, \\ \{0,1\}, \text{якщо } a_i \in \{\bot,\top\}. \end{cases} \quad (8)$$

Зазначимо, що $C \subseteq f_c(f_a(C))$ для будь-якого $C \in 2^{V_n}$, тобто під час використання абстракції не відбувається втрати конкретних значень.

На рис. 1 представлена модель комірки пам’яті, яка не потребує ініціалізації.

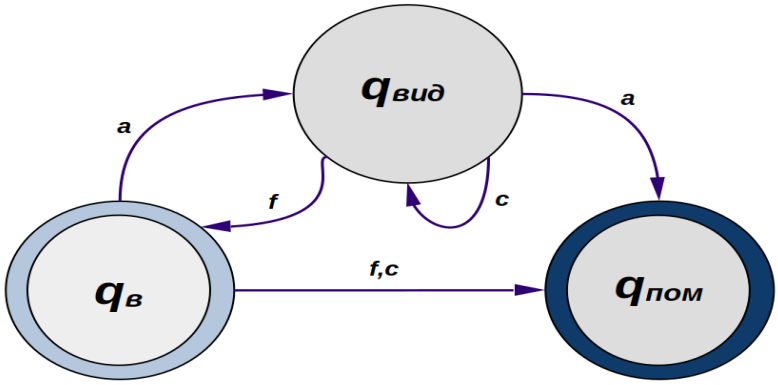

Рис. 1. Модель комірки пам’яті, яка не потребує ініціалізації
Джерело: розроблено автором в середовищі LibreOffice (знімок з екрану)

Така модель дозволяє зберігати дані в пам’яті без необхідності попереднього визначення значень, що можуть бути використані для збереження тимчасових або робочих даних. Це забезпечує більш гнучке і ефективне використання пам’яті, зокрема в умовах високонавантажених обчислювальних систем. Комірка пам’яті в цій моделі може бути автоматично оновлена або змінена в залежності від процесів, що виконуються, без ризику виникнення помилок через відсутність ініціалізації. Однак важливо забезпечити механізми контролю доступу до таких ділянок пам’яті для запобігання несанкціонованому доступу чи модифікаціям, які можуть призвести до порушення цілісності даних або атак [1], [11] – [13].

Комірка може перебувати в одному з трьох станів: $q_{\text{в}}$ — комірка вільна, $q_{\text{вид}}$ — комірка виділена, $q_{\text{пом}}$ — помилка. При переході в стан $q_{\text{пом}}$ відбувається зупинка роботи моделі. Перехід між цими станами здійснюється за допомогою операцій, позначених у моделі: a — відповідає за виділення пам’яті для комірки, f — за її звільнення (після того, як комірка більше не потрібна для зберігання даних), c — позначає звернення до даних, що зберігаються в цій комірці. Важливою особливістю цієї моделі є визначення того, чи потрібно захищати комірки пам’яті від небажаних звернень, зокрема від несанкціонованого читання. Залежно від цього, модель розрізняє два варіанти роботи: у разі потреби захисту виконуватиметься лише запис в комірку, а у разі відсутності потреби в захисті — можуть бути дозволені як операції читання, так і запису. Це дозволяє моделювати різні аспекти управління пам’яттю в Grid-середовищах, де критично важливе забезпечення безпеки даних під час виконання обчислювальних завдань [4] – [8]. Зокрема, для забезпечення захисту від шкідливого коду необхідно вчасно виявляти та контролювати помилки або неналежні операції з пам’яттю, що можуть призвести до витоку або пошкодження даних [14].

Згідно з моделлю (рис. 2), представлено кілька ключових етапів взаємодії з комірками пам’яті, кожен з яких має важливе значення для забезпечення безпеки в обчислювальних середовищах, таких як Grid-системи.

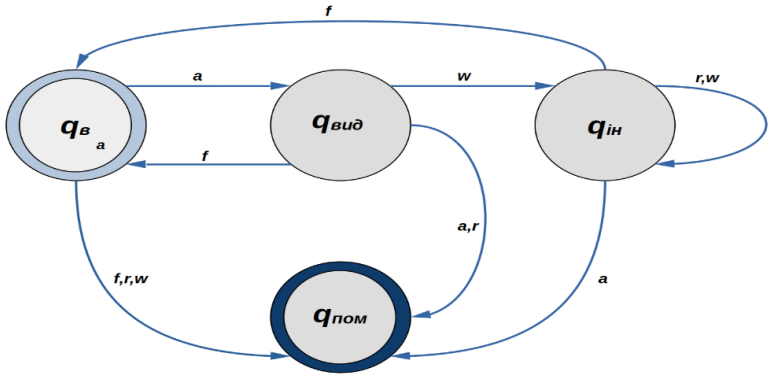

Рис. 2. Модель комірки пам'яті, яка потребує ініціалізації
Джерело: розроблено автором в середовищі LibreOffice (знімок з екрану)

Модель комірки пам'яті, яка потребує ініціалізації, відрізняється від попередньої тим, що перед її використанням необхідно ініціалізувати значення вмісту комірки, щоб забезпечити коректність її функціонування. Якщо комірка не була ініціалізована перед використанням, або спроба звернення до неї здійснюється в її некоректному стані, відбувається перехід до стану $q_{\text{пом}}$, що призводить до зупинки виконання задачі і сигналізує про помилку в системі. Це забезпечує більшу безпеку і стабільність роботи системи, знижуючи ймовірність появи уразливостей, пов'язаних з неконтрольованими зверненнями до неініціалізованих або пошкоджених осередків пам'яті [5], [11]. Таким чином, модель комірки пам'яті, що потребує ініціалізації, є ефективним інструментом для захисту від помилок у пам'яті і шкідливих програмних впливів, забезпечуючи більшу надійність та безпеку в процесах обробки даних у Grid-середовищах.

Визначення чотирьох можливих станів комірки пам'яті — $q_{\text{в}}$ (вільна), $q_{\text{вид}}$(виділена, але не ініціалізована), $q_{\text{пом}}$ (помилка), $q_{\text{ін}}$ (ініціалізована) — дає змогу чітко відстежувати та керувати її станом під час виконання обчислювальних завдань. Важливим аспектом є те, що початковий стан $q_{\text{в}}$ забезпечує можливість для подальшого виділення пам'яті для обчислень, тоді як зміна на $q_{\text{пом}}$ вказує на критичну помилку, що може зупинити виконання моделі [6].

Перехід між різними станами моделі здійснюється через чітко визначені операції: a — виділення пам'яті, f — звільнення пам'яті, r — читання з комірки та w — запис у комірку. Кожна з цих операцій не тільки виконує відповідні функції, але й служить для контролю за правильністю обробки даних у пам'яті, тим самим забезпечуючи запобігання можливим атакам або несанкціонованому доступу. Процес моделювання комірки пам'яті, що потребує ініціалізації, може бути адаптований до моделі без потреби ініціалізації, шляхом заміни операції a на послідовність операцій a, w, що виконуються по черзі. Це дозволяє реалізувати захист пам'яті без додаткових складнощів, зберігаючи при цьому ефективність і безпеку. В такому контексті, зміни в логіці роботи з пам'яттю можуть значно підвищити стійкість системи до шкідливого коду, що є критичним аспектом для забезпечення захисту даних в Grid-середовищах [1], [2], [13] – [15]. Запровадження такої моделі дозволяє знизити ризик атак, спрямованих на зміну вмісту пам'яті, та підвищує стійкість до помилок у роботі з обчислювальними завданнями. Застосування подібної стратегії є важливим елементом методики захисту Grid-середовища від шкідливого коду, особливо в умовах, коли необхідно забезпечити безперервність і цілісність виконуваних обчислювальних завдань в умовах великої кількості користувачів і ресурсів [2].

При статичному аналізі формальних властивостей та моніторингу обчислювальних завдань необхідно здійснювати контроль за відповідністю введених обчислювальних завдань моделі комірки пам'яті, яка повинна відображати вимоги політики безпеки, забезпечуючи тим самим коректність і безпеку виконання задач у середовищі обчислень. Це включає перевірку кожної операції з пам'яттю на її відповідність передбаченим стандартам, таким як обмеження на доступ до не ініціалізованих областей пам'яті, а також контроль за виконанням операцій читання та запису згідно з визначеними правами доступу [3]. Також важливо проводити моніторинг виконання обчислювальних завдань у реальному часі для виявлення будь-яких аномалій або порушень, що можуть свідчити про спроби несанкціонованого доступу або маніпуляцій з даними. Це дозволяє своєчасно виявити потенційні загрози або помилки в системі, які можуть призвести до порушення політики безпеки або навіть до втрати даних. Забезпечення відповідності обчислювальних завдань цій моделі пам'яті дозволяє не лише підвищити рівень безпеки, але й знизити ймовірність появи уразливостей, пов'язаних з некоректною роботою з пам'яттю, таких як переповнення буфера або неконтрольовані зміни даних. Таким чином, постійний моніторинг і аналіз виконуваних завдань є важливим елементом загальної стратегії безпеки в Grid-середовищах, що дозволяє забезпечити належний рівень захисту від шкідливих впливів та помилок [4].

У процесі взаємодії обчислювальних завдань з диспетчером системи розподілених обчислень (СРВ) важливо запобігти реалізації атак типу «відмова в обслуговуванні», що можуть спричинити відмову в обробці запитів або порушення нормальної роботи диспетчера. Для більш детального опису ситуації введемо наступні позначення: V — пропускна здатність каналу або система обробки даних диспетчера СРВ; U — швидкість передачі даних, що забезпечується обчислювальним вузлом; N — загальна кількість обчислювальних вузлів у СРВ; W — загальний обсяг даних, що передаються диспетчеру СРВ всіма обчислювальними вузлами за одиницю часу. Атака типу «відмова в обслуговуванні» вважається успішно реалізованою, коли обсяг переданих даних (W) значно перевищує пропускну здатність каналу (V). Однак таку атаку неможливо реалізувати, якщо для кожного обчислювального вузла виконується умова $U < V/N$, оскільки при цьому $W<V$, що гарантує безпеку роботи мережі [6]. Здійснення моніторингу мережевого взаємодії, який проводиться як на диспетчері СРВ, так і на обчислювальних вузлах, дозволяє своєчасно виявляти й запобігати передачі даних з швидкістю, що перевищує визначену норму (U), і таким чином захищати диспетчер СРВ від атак типу «відмова в обслуговуванні», спричинених шкідливими обчислювальними завданнями.

Процес побудови концептуальної та функціональної моделей системи захисту системи розподілених обчислень (СРВ) від шкідливого програмного забезпечення (ПО) передбачає розробку архітектури, здатної забезпечити високий рівень безпеки в умовах складної інфраструктури. Зважаючи на специфіку захищуваного середовища, яке включає численні обчислювальні вузли, а також гетерогенне апаратне й програмне забезпечення, визначено основні вимоги до системи захисту СРВ від шкідливого коду, який може бути виконаний під час обчислень. Зокрема, система захисту повинна: 1) Виявляти шкідливий код та запобігати його виконанню, мінімізуючи потенційні загрози для безпеки [1], [2]; 2) Виявляти спроби реалізації атак типу «відмова в обслуговуванні» на диспетчер СРВ і вжити відповідних заходів для їх запобігання [3], [4]; 3) Функціонувати в умовах гетерогенного апаратного та програмного середовища, забезпечуючи сумісність з різними платформами та компонентами; 4) Мати мінімальний вплив на продуктивність системи, зменшуючи додаткове навантаження на вузли СРВ під час виконання обчислень [5], [6]; 5) Працювати в автоматичному режимі, забезпечуючи

безперервний захист без необхідності ручного втручання [6], [7]. Система захисту СРВ від шкідливого ПЗ складається з безпечної бібліотеки, модуля динамічної верифікації та верифікатора вихідного коду, а також передбачає інтерфейс для розробників програмного забезпечення. Вхідними даними для системи є вихідний код обчислювального завдання, написаного мовою С.

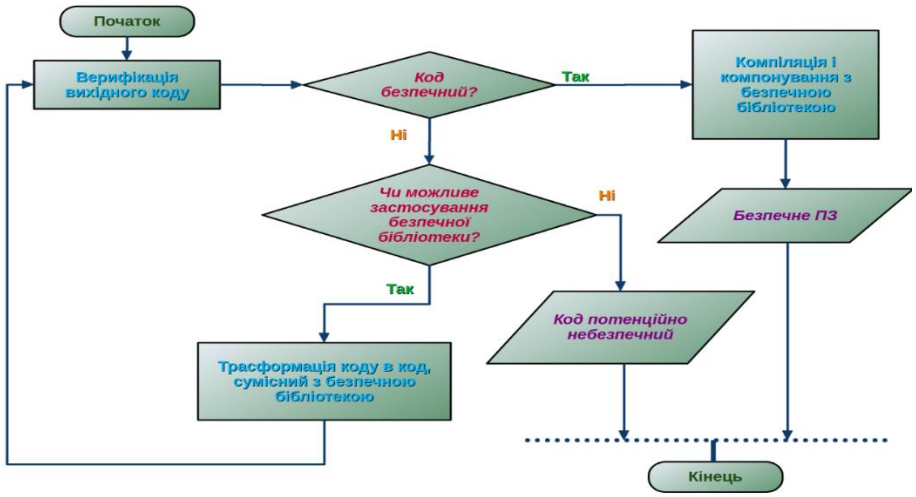

Рис. 3. Алгоритм перевірки безпеки коду програм
Джерело: розроблено автором в середовищі LibreOffice (знімок з екрану)

В автоматичному режимі здійснюється перевірка безпеки цього коду, що, за потреби, включає перетворення потенційно небезпечних ділянок у безпечний код за допомогою безпечної бібліотеки. Алгоритм перевірки безпеки програмного коду, що застосовується в системі, передбачає виконання абстрактної інтерпретації, що дозволяє ефективно виявляти вразливості та забезпечити захист від шкідливих програм і атак під час виконання обчислювальних завдань. Це знижує ризики та підвищує загальний рівень безпеки системи, дозволяючи підтримувати високий рівень захисту в розподілених обчислювальних середовищах.

Перетворення потенційно небезпечного коду на безпечний, що використовує безпечну бібліотеку, виконується в повністю автоматичному режимі. Для цього відбувається заміна потенційно небезпечних операцій на виклики функцій монітора звернень, що забезпечує безпечне виконання таких операцій [7], [16] – [20], [24]. На рис. 4 наведено загальну схему взаємодії обчислювального завдання з безпечною бібліотекою.

Рис. 4. Загальна схема взаємодії з безпечною бібліотекою
Джерело: розроблено автором в середовищі LibreOffice (знімок з екрану)

Схема демонструє основні етапи обробки обчислювальних завдань у Grid-середовищі за допомогою безпечної бібліотеки. Взаємодія починається з прийому вихідного коду завдання, який автоматично перевіряється на наявність потенційно небезпечних ділянок. У разі виявлення таких ділянок здійснюється їх трансформація в виклики функцій безпечної бібліотеки. Безпечна бібліотека виступає проміжним рівнем між обчислювальним завданням та апаратними чи програмними ресурсами системи. Її функції включають контроль за коректністю виконання операцій, виявлення спроб несанкціонованого доступу та блокування потенційно небезпечних дій. Крім того, бібліотека дозволяє реалізувати політики безпеки, адаптовані до специфіки Grid-середовища, такі як динамічна перевірка прав доступу, моніторинг викликів функцій і їх відповідність встановленим правилам. Використання такої схеми забезпечує ефективний захист від шкідливого коду та підвищує загальну стійкість системи до кіберзагроз [8], [9], [16].

Такий підхід дозволяє автоматизувати процес захисту коду та зменшити ризики, пов’язані з людським фактором, що є важливим для забезпечення надійності роботи Grid-середовища. Монітор звернень перевіряє коректність викликів функцій, аналізує дотримання політик безпеки та, у разі необхідності, блокує виконання операцій, що можуть становити загрозу. Це забезпечує не тільки захист від шкідливого коду, але й контроль за небажаними поведінковими аномаліями обчислювальних завдань. Застосування безпечних бібліотек є ключовим елементом сучасних підходів до захисту розподілених обчислювальних середовищ, оскільки вони забезпечують сумісність з різними програмними платформами та мінімізують ризики, пов’язані з впровадженням шкідливого коду в процес виконання завдань [10] – [12].

Безпечна бібліотека в системах захисту Grid-середовища виконує ключові функції моніторингу звернень та обробки операцій з пам’яттю та вводу-виводу. Основна мета такої бібліотеки — мінімізувати ризики виконання шкідливого коду та запобігти потенційним загрозам безпеці під час виконання обчислювальних завдань [13]. Безпека взаємодії обчислювальних завдань із загальним пулом пам’яті перевіряється методами статичної верифікації, що дозволяє виявляти можливі вразливості до виконання завдань. Захищений пул пам’яті додатково контролюється монітором звернень, який здійснює перевірку всіх параметрів, що використовуються під час роботи із пам’яттю. Монітор також забезпечує перевірку коректності взаємодії зі службами системи розподілених обчислень (СРВ) та операційної системи (ОС). Таким чином, уся система захищена від несанкціонованого доступу до ресурсів та інших потенційних загроз [14] – [17].

Політика безпеки, що визначає правила роботи системи, включає такі параметри, як обов’язкове обнулення виділеної пам’яті, захист від несанкціонованого читання довільних областей пам’яті, заборона на виклик довільних функцій ОС та асемблерних команд, а також визначення дозволених імен і режимів доступу до файлів ОС. Крім цього, політика вимагає ведення аудиту дій програмного забезпечення та захисту від атак типу «відмова в обслуговуванні» на диспетчер СРВ [19], [20]. Актуальна політика безпеки формується шляхом об’єднання вимог, встановлених власниками диспетчера СРВ та обчислювального вузла. У процесі інтеграції обираються найбільш суворі обмеження з наявних політик, що забезпечує максимальний рівень захисту [22] – [24]. У випадку використання різних платформ СРВ та ОС реалізуються відповідні версії монітора звернень, які забезпечують підтримку безпечної взаємодії з окремими службами кожної платформи. Це дозволяє залишити інші елементи системи незмінними, забезпечуючи адаптивність і масштабованість рішення.

Сучасні методики захисту також враховують можливість автоматизованого налаштування політик безпеки на основі аналізу поведінки завдань. Такий підхід дозволяє

системі оперативно реагувати на нові загрози, мінімізуючи вплив людського фактора і забезпечуючи стабільну роботу навіть у гетерогенних обчислювальних середовищах. Завдяки цьому Grid-середовища стають більш захищеними, зберігаючи високу продуктивність і надійність під час виконання обчислювальних завдань [1], [19] – [21].

Для побудови системи захисту Grid-середовища від шкідливого коду обчислювальних завдань використано сучасні технології та розроблено адаптивні структури даних. Реалізація системи включала створення механізмів статичної та динамічної верифікації коду, аналізу потенційно небезпечних операцій, а також оцінювання ефективності та продуктивності запропонованого рішення. Основним інструментом розробки є мова програмування C#, яка забезпечує високу продуктивність, оптимальне управління пам’яттю та можливість інтеграції з іншими програмними компонентами. Для створення верифікатора використовувалися генератори лексичних аналізаторів Lex і синтаксичних аналізаторів Bison, які гарантують точність обробки коду. Для розробки інтерфейсу програмування обчислювальних завдань застосовувалася мова C#, що спрощує інтеграцію з користувацькими програмами [7]. Для оцінки продуктивності розробленого безпечного коду було виконано серію тестів, які включали як спеціально створені сценарії, так і загальнодоступні програми. Тести поділялися на три основні категорії: тип A — програми, що активно працюють з вказівниками; тип B — програми, які інтенсивно виділяють та звільняють пам’ять; тип C — програми, орієнтовані на виконання складних математичних обчислень. Завдяки використанню сучасних підходів і технологій, розроблена методика забезпечує ефективний захист Grid-середовища від шкідливого коду, мінімізуючи ризики порушення безпеки та зберігаючи високу продуктивність обчислювальних завдань [3], [4], [21] – [23],[25].

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У результаті проведеного дослідження методики захисту Grid-середовища від шкідливого коду під час виконання обчислювальних завдань, були виявлені основні проблеми, пов’язані з безпекою в таких системах, зокрема зростаюча кількість загроз, зокрема поширення шкідливого коду, що може серйозно порушити нормальну роботу та функціонування системи. Grid-середовища, які об’єднують географічно розподілені ресурси для вирішення складних обчислювальних завдань, є важливими не лише для наукових досліджень, але й для багатьох промислових та бізнесових додатків. Тому забезпечення їх безпеки є першочерговим завданням для забезпечення безперервної роботи таких систем.

Однією з ключових проблем є масштабність та різнорідність ресурсів, що використовуються в Grid-середовищах. Ці особливості потребують застосування спеціалізованих методів захисту, які здатні ефективно працювати з розподіленими і часто гетерогенними обчислювальними платформами. Одним з важливих аспектів є впровадження програмно-апаратних рішень для моніторингу і блокування шкідливого коду. Ці рішення дозволяють виявляти загрози на ранніх етапах і ефективно запобігати їх поширенню, зберігаючи високу продуктивність системи. Особлива увага в дослідженні приділена моделі управління пам’яттю в розподілених обчислювальних середовищах. Правильна ініціалізація, виділення та звільнення пам’яті є важливими елементами для забезпечення цілісності даних і запобігання несанкціонованому доступу до них. Розроблена модель взаємодії комірок пам’яті в Grid-середовищах дозволяє ефективно управляти доступом до пам’яті, що є критичним для запобігання шкідливим впливам, таким як переповнення буферів або спроби несанкціонованого доступу до даних.

Запропоновані методи аутентифікації та моніторингу також відіграють важливу роль у захисті пам'яті і системи загалом. Вони дозволяють оперативно виявляти та блокувати атаки, які можуть бути спричинені шкідливим кодом. Використання таких методів дозволяє мінімізувати ризики втручання шкідливих програм і підвищити стійкість системи до зовнішніх загроз.

З огляду на постійний розвиток технологій, перспективи подальших досліджень мають велике значення для подальшого удосконалення методів захисту Grid-середовищ. В першу чергу, необхідно зосередити увагу на оптимізації взаємодії між безпечними бібліотеками та обчислювальними завданнями, що виконуються в гетерогенних платформах. Це дозволить зменшити навантаження на систему та підвищити її ефективність, не знижуючи рівень безпеки. Одним з напрямків є також розвиток автоматизованих методів виявлення шкідливого коду, зокрема на основі поведінкового аналізу. Цей підхід дозволить ефективно прогнозувати нові загрози та швидко реагувати на них, що є важливим для адаптивності системи. Інтеграція з новими технологіями, такими як квантові обчислення та штучний інтелект, відкриває нові можливості для підвищення безпеки. Квантові обчислення можуть суттєво змінити підхід до захисту даних, а штучний інтелект дозволяє створювати системи, які зможуть прогнозувати та виявляти потенційні атаки в реальному часі.

Розробка універсальних методів захисту, які працюватимуть в різнорідних середовищах, де використовуються різні апаратні та програмні платформи, є ще одним важливим напрямком. Такі методи повинні забезпечувати високу ефективність захисту, враховуючи специфіку обчислювальних завдань і платформ, на яких вони виконуються. Крім того, удосконалення міжнародних стандартів безпеки, таких як ISO/IEC 27001, для підтримки високого рівня довіри до Grid-середовищ буде сприяти їх широкому впровадженню у різних сферах.

Загалом, подальші дослідження в галузі захисту Grid-середовищ від шкідливого коду дозволять значно підвищити їхню надійність і безпеку. Це відкриє нові можливості для застосування таких середовищ у різних галузях, таких як наука, медицина, промисловість та бізнес, де високий рівень безпеки є критично важливим для успішного функціонування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sharma, R., & Shahi, M. (2019). Neural Network Models for Cyber Risk Assessment. *Springer*.
2. Chen, M., & Zhang, H. (2020). Integrating Fuzzy Logic in Information Security Risk Assessment. *Journal of Risk Analysis, 31(4),* 202–211.
3. Gupta, R., & Soni, A. (2020). Evaluation of Risk Factors in Information Security Using Fuzzy Logic. *International Journal of Computer Science and Information Security, 18(4),* 191–198.
4. Li, Z., & Yang, J. (2021). A Comparative Study of Risk Management Frameworks in Cybersecurity. *Journal of Cybersecurity Research, 8(2),* 157–168.
5. Wang, P., & Chen, H. (2021). Fuzzy Cognitive Maps in Information Risk Management. *International Journal of Intelligent Systems, 36(4),* 1234–1252.
6. Ravichandran, V., & Kumar, A. (2021). Cybersecurity Risk Management and Mitigation Strategies. *Elsevier.*
7. Bensoussan, A. M., & Green, M. D. (2018). Security in Grid computing environments. *IEEE Transactions on Cloud Computing, 6(4),* 976–989.
8. Curtis, R. M., & Chen, M. J. (2020). Risk Management in Information Security for Distributed Systems. *Journal of Computer Security, 28(1),* 65–77.
9. Chen, A. P., & Zhang, S. C. (2019). A secure Grid computing system design using cryptographic techniques. *International Journal of Network Security, 17(3),* 301–315.

10. Zhang, X., Liu, L., & Zhou, Y. (2021). Grid Security: A Comprehensive Survey. *Future Generation Computer Systems*, *109*, 156–169.
11. Tan, L. M., Zhou, T. B., & Song, W. X. (2020). Enhancing security in grid computing systems using adaptive trust models. *Journal of Grid Computing*, *18(2)*, 237–249.
12. Wang, T., Chen, S. H., & Zhao, L. H. (2020). A survey on grid security and related frameworks. *Computers & Security*, *96*.
13. Li, J., & Zhang, Y. (2019). Intrusion detection and prevention in Grid computing systems. *International Journal of Information Security*, *18(1)*, 29–41.
14. Kostiuk, Y. V., & Voytkevich, A. A. (2024). Research on Technologies for Detection and Identification of Intruders for Corporate Network Protection. *"Science and Technology Today" (Series "Pedagogy", Series "Law", Series "Economics", Series "Physical and Mathematical Sciences", Series "Engineering"), 4(32),* 1017–1032.
15. Yang, A. P., Liu, Y. S., & Xu, J. H. (2020). Grid Computing: Security challenges and solutions. *Computer Science and Technology*, *34(1)*, 33–45.
16. Zhao, L. L., & Li, J. T. (2021). Protection techniques for grid computing systems against malicious code. *Journal of Computer Networks and Communications*.
17. Kostiuk, Y. V., & Shapran, V. O. (2024). Technologies for Detection of Anomalous Events and Signatures in Real-Time. *"Science and Technology Today" (Series "Pedagogy", Series "Law", Series "Economics", Series "Physical and Mathematical Sciences", Series "Engineering"*), *4(32)*, 1069–1084.
18. Xu, P., Zhang, L., & Chen, J. (2022). The impact of malicious code on distributed systems and countermeasures. *Journal of Information Security*, *41(2)*, 156–170.
19. Dovzhenko, N., Mazur, N., Kostiuk, Y., & Rzaieva, S. (2024). Integration of IoT and Artificial Intelligence into Intelligent Transportation Systems. *Electronic Scientific Journal "Cybersecurity: Education, Science, Technology", 2(26),* 430–444.
20. Kryvoruchko, O., Kostiuk, Y., & Desiatko, A. (2024). Systematization of signs of unauthorized access to corporate information based on application of cryptographic protection methods. *Ukrainian Scientific Journal of Information Security*, *30(1)*, 140–149.
21. Li, Q., & van der Schaar, M. (2004). Providing adaptive QoS to layered video over wireless local area networks through realtime retry limit adaptation. *IEEE Transactions on Multimedia*, *6(2)*, 278–290.
22. Kostiuk, Y., Bebeshko, B., Kryuchkova, L., Litvinov, V., Oksanych, I., Skladannyi, P., & Khorolska, K. (2024). Information Protection and Data Exchange Security in Wireless Mobile Networks with Authentication and Key Exchange Protocols. *Electronic Professional Scientific Journal "Cybersecurity: Education, Science, Technology", 1(25),* 229–252.
23. Shen, H. P., Li, F., & Zhang, Z. (2021). AI-based approaches for intrusion detection in Grid computing. *IEEE Transactions on Artificial Intelligence*, *5(3)*, 256–270.
24. Zhang, Y. T., & Zhou, S. H. (2020). Risk assessment models for secure execution of computational tasks in grid environments. *Journal of Cybersecurity Research*, *7(4)*, 45–58.
25. Hulak, H. M., Zhiltsov, O. B., Kyrychok, R. V., Korshun, N. V., & Skladannyi, P. M. (2024). *Information and cyber security of the enterprise. Textbook.* Lviv: Publisher Marchenko T. V.

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (27), 2025
ISSN 2663 - 4023

Yuliia Kostiuk
PhD in Computer Science
Associate Professor of the Department of Information and
Cyber Security named after Professor Volodymyr Buriachok
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID ID: 0000-0001-5423-0985
y.kostiuk@kubg.edu.ua

Nadiia Dovzhenko
PhD, Associate Professor,
Associate Professor of the Department of Information and
Cyber Security named after Professor Volodymyr Buriachok
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID ID:0000-0003-4164-0066
nadezhdadovzhenko@gmail.com

Nataliia Mazur
PhD, Associate Professor
Associate Professor of the Department of Information and
Cyber Security named after Professor Volodymyr Buriachok
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID ID: 0000-0001-7671-8287
n.mazur@kubg.edu.ua

Pavlo Skladannyi
PhD, Associate Professor, Head of the Department of Information and
Cyber Security named after Professor Volodymyr Buryachok
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID ID: 0000-0002-7775-6039
p.skladannyi@kubg.edu.ua

Rzaeva Svitlana
Candidate of Technical Sciences, Associate Professor,
Associate Professor of the Department of Computer Science
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID ID: 0000-0002-7589-2045
s.rzaieva@kubg.edu.ua

THE METHODOLOGY FOR PROTECTING GRID ENVIRONMENTS FROM MALICIOUS CODE DURING THE EXECUTION OF COMPUTATIONAL TASKS

Abstract. The article is dedicated to the protection of Grid environments from malicious code that can be integrated during the execution of computational tasks. The main issues are identified, including high risks of malicious interference that could disrupt the operation of computational nodes or paralyze the entire system. The problem of spreading malicious code, which can disable nodes, is considered separately. A protection methodology based on automatic source code verification is proposed, allowing the detection of harmful programs before execution without significant impact on performance. Approaches to threat detection, including static and dynamic code analysis, as well as the use of machine learning algorithms for attack prediction, are discussed. The integration of such methodologies will enhance the security of Grid environments and promote their application in science and industry. The memory management model describes cells that can be in the states of free, allocated, or erroneous, and considers the need to protect memory from unauthorized access. Transitions between states are made through defined operations that ensure data security. The protection architecture includes a secure library, a dynamic verification module, and an access monitor, allowing effective monitoring and protection of the system from malicious software. The security strategy is based on adaptive policy management and static verification to prevent threats, enhancing resilience to cyber threats in real time.

Keywords: Grid environment; malicious code; risks of malicious interference; code verification; static and dynamic analysis; machine learning; memory security; protection architecture; adaptive security policy management.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Sharma, R., & Shahi, M. (2019). Neural Network Models for Cyber Risk Assessment. *Springer*.
2. Chen, M., & Zhang, H. (2020). Integrating Fuzzy Logic in Information Security Risk Assessment. *Journal of Risk Analysis*, *31(4)*, 202–211.
3. Gupta, R., & Soni, A. (2020). Evaluation of Risk Factors in Information Security Using Fuzzy Logic. *International Journal of Computer Science and Information Security*, *18(4)*, 191–198.
4. Li, Z., & Yang, J. (2021). A Comparative Study of Risk Management Frameworks in Cybersecurity. *Journal of Cybersecurity Research*, *8(2)*, 157–168.
5. Wang, P., & Chen, H. (2021). Fuzzy Cognitive Maps in Information Risk Management. *International Journal of Intelligent Systems*, *36(4)*, 1234–1252.
6. Ravichandran, V., & Kumar, A. (2021). Cybersecurity Risk Management and Mitigation Strategies. *Elsevier*.
7. Bensoussan, A. M., & Green, M. D. (2018). Security in Grid computing environments. *IEEE Transactions on Cloud Computing*, *6(4)*, 976–989.
8. Curtis, R. M., & Chen, M. J. (2020). Risk Management in Information Security for Distributed Systems. *Journal of Computer Security*, *28(1)*, 65–77.
9. Chen, A. P., & Zhang, S. C. (2019). A secure Grid computing system design using cryptographic techniques. *International Journal of Network Security*, *17(3)*, 301–315.
10. Zhang, X., Liu, L., & Zhou, Y. (2021). Grid Security: A Comprehensive Survey. *Future Generation Computer Systems*, *109*, 156–169.
11. Tan, L. M., Zhou, T. B., & Song, W. X. (2020). Enhancing security in grid computing systems using adaptive trust models. *Journal of Grid Computing*, *18(2)*, 237–249.
12. Wang, T., Chen, S. H., & Zhao, L. H. (2020). A survey on grid security and related frameworks. *Computers & Security*, *96*.
13. Li, J., & Zhang, Y. (2019). Intrusion detection and prevention in Grid computing systems. *International Journal of Information Security*, *18(1)*, 29–41.
14. Kostiuk, Y. V., & Voytkevich, A. A. (2024). Research on Technologies for Detection and Identification of Intruders for Corporate Network Protection. *"Science and Technology Today" (Series "Pedagogy", Series "Law", Series "Economics", Series "Physical and Mathematical Sciences", Series "Engineering")*, *4(32)*, 1017–1032.
15. Yang, A. P., Liu, Y. S., & Xu, J. H. (2020). Grid Computing: Security challenges and solutions. *Computer Science and Technology*, *34(1)*, 33–45.
16. Zhao, L. L., & Li, J. T. (2021). Protection techniques for grid computing systems against malicious code. *Journal of Computer Networks and Communications*.
17. Kostiuk, Y. V., & Shapran, V. O. (2024). Technologies for Detection of Anomalous Events and Signatures in Real-Time. *"Science and Technology Today" (Series "Pedagogy", Series "Law", Series "Economics", Series "Physical and Mathematical Sciences", Series "Engineering"*), *4(32)*, 1069–1084.
18. Xu, P., Zhang, L., & Chen, J. (2022). The impact of malicious code on distributed systems and countermeasures. *Journal of Information Security*, *41(2)*, 156–170.
19. Dovzhenko, N., Mazur, N., Kostiuk, Y., & Rzaieva, S. (2024). Integration of IoT and Artificial Intelligence into Intelligent Transportation Systems. *Electronic Scientific Journal "Cybersecurity: Education, Science, Technology"*, *2(26)*, 430–444.
20. Kryvoruchko, O., Kostiuk, Y., & Desiatko, A. (2024). Systematization of signs of unauthorized access to corporate information based on application of cryptographic protection methods. *Ukrainian Scientific Journal of Information Security*, *30(1)*, 140–149.
21. Li, Q., & van der Schaar, M. (2004). Providing adaptive QoS to layered video over wireless local area networks through realtime retry limit adaptation. *IEEE Transactions on Multimedia*, *6(2)*, 278–290.
22. Kostiuk, Y., Bebeshko, B., Kryuchkova, L., Litvinov, V., Oksanych, I., Skladannyi, P., & Khorolska, K. (2024). Information Protection and Data Exchange Security in Wireless Mobile Networks with Authentication and Key Exchange Protocols. *Electronic Professional Scientific Journal "Cybersecurity: Education, Science, Technology"*, *1(25)*, 229–252.

23. Shen, H. P., Li, F., & Zhang, Z. (2021). AI-based approaches for intrusion detection in Grid computing. *IEEE Transactions on Artificial Intelligence*, *5(3)*, 256–270.
24. Zhang, Y. T., & Zhou, S. H. (2020). Risk assessment models for secure execution of computational tasks in grid environments. *Journal of Cybersecurity Research, 7(4)*, 45–58.
25. Hulak, H. M., Zhiltsov, O. B., Kyrychok, R. V., Korshun, N. V., & Skladannyi, P. M. (2024). *Information and cyber security of the enterprise. Textbook.* Lviv: Publisher Marchenko T. V.