DOI 10.31891/2307-5732-2025-347-76
УДК 004.75

РЕГІДА ПАВЛО
Хмельницький національний університет
https://orcid.org/0000-0002-6591-7069
email: pavlo.rehida@gmail.com

МЕТОД ОРГАНІЗАЦІЇ РОЗПОДІЛЕНОЇ СИСТЕМИ ВИЯВЛЕННЯ ІНФІКОВАНИХ ПРОГРАМ В ІЗОЛЬОВАНИХ СЕРЕДОВИЩАХ

Цю роботу присвячено дослідженню методів і засобів організації розподілених систем для виявлення інфікованих програм, що використовують техніки уникнення виявлення. Для проведення дослідження застосовується множина ізольованих середовищ для аналізу їхньої поведінки під час виконання. Актуальність дослідження та аналізу, а також пошук нових методів виявлення зловмисного програмного забезпечення є надзвичайно важливою. Це підтверджується великою кількістю нових екземплярів та швидким розвитком методів уникнення виявлення. Незважаючи на велику кількість різноманітних засобів виявлення таких загроз, щороку фіксуються численні випадки компрометації персональних та корпоративних даних на різних веб-ресурсах та платформах. Складність виявлення таких загроз пов'язана, зокрема, з використанням методів, що містять поліморфні властивості, які ускладнюють процес виявлення. У статті проведено аналіз застосування технологій розподілених систем для пошуку екземплярів зловмисного програмного забезпечення та представлено передові рішення. Особливу увагу приділено перевагам використання грід-обчислювальних систем і особливостям їхньої організації. Мета цієї статті – представити метод, що застосовує розподіл завдань між елементами грід-обчислювальної системи для ефективного розподілу навантаження з урахуванням автономності обчислювальних елементів. У ролі завдань виступають запити до обчислювальних елементів на виконання та аналіз інфікованих програм, що дозволяє сформувати їхню поведінкову модель. Запропонована система використовує центральний сервер для координації всіх внутрішніх процесів комунікації, збору результатів виконання інфікованих програм та їх подальший аналіз для остаточного визначення наявності зловмисної активності у програмі. Запропонований метод дає змогу не лише організувати розподіл завдань, а й зробити його оптимальним, зважаючи на автономність елементів, використовуючи розроблений мережевий протокол взаємодії центрального сервера та обчислювальних елементів.

Ключові слова: розподілені системи, інфікована програма, розподілення завдань

REHIDA PAVLO
Khmelnytskyi National University

METHOD OF ORGANIZING A DISTRIBUTED SYSTEM FOR DETECTING INFECTED PROGRAMS IN ISOLATED ENVIRONMETNS

This work is dedicated to the study of methods and tools for organizing distributed systems to detect infected programs that utilizes evasion techniques. The research employs a set of isolated environments to analyse program behaviour during execution. The relevance of research and analysis, as well as the search for new methods of detecting malicious software, has a primary importance. This is confirmed by the large number of new malware samples and the rapid development of evasion techniques. Despite the availability of various detection tools, numerous cases of personal and corporate data breaches are recorded annually on different web resources and platforms. The complexity of detecting such threats is particularly associated with the use of methods that incorporated polymorphic features, making detection significantly more challenging. This paper analyses the application of distributed system technologies for identifying of malicious software and presents modern solutions. Special attention is given to the advantages of using grid computing systems and their organizational specifics. The aim of this study is to introduce a method that utilizes task distribution among elements of a grid computing system to efficiently balance the load while considering the autonomy of computing elements. Tasks in this context involve sending requests to computing elements for executing and analysing infected programs, enabling the formation of behavioural models. The proposed system employs a central server to coordinate all internal communication processes, collect execution results of infected programs, and further analyse them to detect the presence of malicious activity in the software. The proposed method not only enables task distribution but also optimizes it by considering the autonomy of system elements. This is achieved through a developed network protocol that facilitates interaction between the central server and computing elements.

Keywords: distributed systems, infected software, task distribution

Постановка проблеми

Численні дослідження та статистичні дані, зібрані науковими лабораторіями та дослідницькими центрами, фіксують зростання загроз [1-3] як за кількістю, та і за рівнем складності. Розробники зловмисного програмного забезпечення активно використовують ці загрози для отримання несанкціонованого доступу до пристроїв користувачів і реалізації різних схем зловмисної діяльності. Для успішного розповсюдження зловмисного програмного забезпечення зловмисники використовують різні методи, зокрема інфікування файлів та програм [4,5], яким користувачі довіряють, і які регулярно використовують. Хоча існує багато засобів захисту, методи вторгнення постійно вдосконалюються. Одним із найскладніших викликів для антивірусних систем є поліморфні віруси – зловмисне програмне забезпечення, що змінює свій код при кожному виконанні або розповсюджені. Такі віруси використовують техніки уникнення виявлення, зокрема емуляцію, динамічне шифрування, мутацію та обфускацію [6], що значно ускладнює їх ідентифікацію [7].

Головною особливістю поліморфних вірусів [8,9] є збереження функціональності при постійній зміні коду, що робить їх практично невидимими для сигнатурного аналізу [10]. Окрім цього, вони можуть інфікувати [11], вбудовуючи свій код і модифікуючи структуру виконуваних файлів. Ще одна важлива

характеристика – змінна поведінка [12]. Поліморфні віруси можуть адаптувати активність залежно від середовища виконання до певного моменту, що залишатися непоміченими. Через це ефективне виявлення таких загроз вимагає розширеного поведінкового аналізу із використанням методів динамічної емуляції. Тому, дослідження актуальних загроз та вдосконалення методів захисту є актуальним завданням сьогодення. Для їх вирішення фахівці активно застосовують напрацювання суміжних ІТ-сфер, зокрема: розподілені системи, аналіз великих даних, хмарні технології, машинне навчання, тощо.

Розподілені системи отримали широке застосування у кібербезпеці. Вони можуть використовуватись як програмні рішення, що об'єднують велику кількість програмних елементів для створення централізованої системи захисту, або як обчислювальні платформи, які забезпечують необхідну обчислювальну потужність для аналізу зловмисного програмного забезпечення.

Аналіз останніх досліджень

Сучасні обчислювальні системи зазнали значного розвитку, і сьогодні існує кілька ключових підходів до їх організації. Серед них особливо виділяються кластерні, грід-обчислювальні та хмарні системи. Кожна з них має свої особливості, переваги та сфери застосування, що залежить від вимог по продуктивності, масштабованості та управління ресурсами. Грід-обчислювальні системи – це розподілені системи, що поєднують автономні обчислювальні потужності. Основною особливістю таких систем є гнучкість, оскільки вони можуть об'єднувати різні типи обчислювальних елементів, незалежно від їх фізичного розташування. На відміну від кластерних систем, де всі обчислювальні ресурси зосереджені в одному центрі, грід-системи залучають ресурси, розподілені по різних географічних локаціях. Це забезпечує високу масштабованість, але також створює додаткові виклики щодо управління ресурсами та балансування навантаження. Хмарні системи мають схожу архітектуру з грід-обчислювальними системами, оскільки обидва підходи використовують розподілені ресурси для виконання обчислювальних завдань. Проте ключова відмінність полягає в тому, що хмарні системи надають ресурси на вимогу, з можливістю динамічного масштабування відповідно до потреб користувача. Натомість грід-обчислювальні системи орієнтовані на ресурсномісткі обчислювальні задачі, де обчислювальні елементи мають певний рівень автономії.

Сучасні аналітичні платформи для виявлення та аналізу зловмисного програмного забезпечення активно використовують розподілені обчислювальні системи. Основними представниками можна виділити: Cuckoo Sandbox, VirusTotal, Hybrid Analysis, ANY.RUN, Joe Sandbox та VMRay Analyzer.

- *Cuckoo Sandbox* – це відкрита платформа для динамічного аналізу підозрілих файлів та програм у ізольованому середовищі. Вона підтримує розгортання аналізу на вузлах розподіленої системи [17], що дозволяє паралельно обробляти велику кількість файлів, підвищуючи ефективність аналізу.
- *VirusTotal* – це хмарний сервіс, що використовує розподілені технології для всебічного сканування підозрілих файлів та програм за допомогою декількох десятків антивірусних рушіїв [18]. Такий підхід дозволяє зіставляти результати сканування з різних джерел, що значно підвищує точність виявлення загроз.
- *Hybrid Analysis* поєднує статичний та динамічний аналіз та використовує розподілену систему для ізольованого запуску файлів [19]. Це дозволяє виявляти складні загрози, які проявляють себе лише під час виконання у реальних умовах.
- *ANY.RUN* – це інтерактивна пісочниця, яка дозволяє користувачам взаємодіяти із інфікованим середовищем у режимі реального часу [20]. Це рішення використовує хмарну інфраструктуру, що дозволяє обробляти множину ізольованих середовищ одночасно.
- *Joe Sandbox* – відноситься до високофункціональних платформ для аналізу зловмисного програмного забезпечення, яка підтримує розподілене виконання таких програм на різних операційних системах [21] (Windows, Linux, macOS, Android). Вона може розгортатись як локально, так і у хмарі.
- *VMRay Analyzer* використовує технологію безагентного моніторингу, яка дозволяє аналізувати поведінку програм під час виконання без втручання в середовище виконання [22]. Такий підхід мінімізує ризик виявлення аналізу зловмисним програмним забезпеченням та ускладнює застосування технік уникнення виявлення.

Завдяки розподіленій архітектурі, ці системи швидко та ефективно обробляти великі обсяги файлів і програм для виявлення зловмисного прояву. Вони є ключовими інструментами у сфері кібербезпеки, оскільки автоматизують та вдосконалюють процес виявлення нових загроз. Крім того, усі ці рішення легко інтегруються в існуючі системи захисту корпоративних користувачів, забезпечуючи додатковий рівень безпеки для організацій.

Архітектура системи аналізу виконання інфікованих програм

Організація грід-обчислювальних систем є ефективним для виконання обчислень, що потребують значних обчислювальних ресурсів. Сучасними реалізаціями таких систем, зокрема, є волонтерські обчислювальні проекти, такі як BOINC, SETI@home та FoldingHome, активне використання яких підтверджує високу ефективність цього підходу на численних прикладах розв'язання наукових задач. Використання грід-обчислень є доцільним, оскільки дає змогу задіяти наявні обчислювальні ресурси без потреби у спеціалізованому обладнанні [23]. Висока надійність грід-систем забезпечується механізмами динамічного балансування навантаження та відмовостійкості інфраструктури. Водночас ці системи потребують реалізації механізмів контролю якості обчислень через автономність

обчислювальних елементів [24]. Для цього застосовують різні методи верифікації результатів, що мінімізують ризики отримання некоректних даних.

У цій роботі представлено метод для централізованої системи, що складається з центрального сервера та множини обчислювальних елементів. Централізована організація визначає характер виконання основних процесів у системі під час її функціонування. Серверний застосунок, базуючись на визначених протоколах мережевої взаємодії, забезпечує коректний порядок роботи, виконуючи такі функції: створення необхідних портів для очікування підключення нових обчислювальних елементів, планування завдань для активних обчислювальних елементів, збір та аналіз отриманих результатів. Таким чином, архітектуру подібної системи можна представити у такому вигляді:

$$System = \{cs, \{ce_1, ce_2, \dots ce_n\}\} \quad (1)$$

де, CS – центральний сервер системи, а $\{ce_1, ce_2, \dots, ce_n\}$ – множина обчислювальних елементів.

Розглянемо усі ключові компоненти елементів системи, які забезпечують необхідну функціональність системи. Центральний сервер містить такі модулі: модулі зовнішньої та внутрішньої комунікації, модуль планування обчислень, модуль довіри та аналітичний модуль. Модуль зовнішньої комунікації використовується для отримання підозрілих файлів або програм від користувачів, а також для передачі результатів аналізу. Модуль внутрішньої комунікації відповідає за взаємодію з обчислювальними елементами, які забезпечують виконання завдань у системі. Модуль планування обчислень фіксує поточну кількість активних обчислювальних елементів і формує завдання для поточної ітерації виконання на основі наявних задач. Модуль довіри розподіляє завдання відповідно до поточних налаштувань безпечних обчислень, приймаючи рішення про необхідність дублювання завдань для підвищення достовірності отриманих результатів. Аналітичний модуль активується за запитом модуля довіри та здійснює порівняння отриманих результатів для виявлення можливої зловмисної активності у проаналізованій програмі.

Обчислювальний елемент містить наступні модулі: модуль внутрішньої комунікації, пісочницю та емулятор. Модуль внутрішньої комунікації забезпечує отримання завдань на виконання, передачу результатів аналізу та сервісної інформації центральному серверу. Крім того, він використовується для опрацювання сервісних запитів, спрямованих на визначення активних і пасивних елементів у мережі. Пісочниця створює ізольоване середовище, яке дозволяє безпечно аналізувати програми та забезпечує необхідний функціонал для проведення аналізу. За допомогою емулятора, відбувається виконання програми в ізольованому середовищі, використовуючи механізми пісочниці. Детальніше, архітектура запропонованої системи розглянута в [25], де також представлено метод виявлення інфікованих програм.

Метод взаємодії та розподілу завдань в запропонованій системі

Для реалізації взаємодії між учасниками системи як серверний застосунок, так і програмне забезпечення обчислювального елемента використовують набір віртуальних потоків. Під час ініціалізації серверного застосунку створюється віртуальний потік, який продовжує працювати протягом усього часу функціонування сервера. Його основне завдання – забезпечення механізму початкового підключення обчислювальних елементів під час кожної сесії взаємодії. Після підключення обчислювального елемента серверний застосунок створює для нього окремий віртуальний потік, який використовується для подальшої комунікації. Використання віртуальних потоків дає змогу створювати велику кількість потоків без значного зниження обчислювальної потужності серверного застосунку. Це забезпечує ефективний розподіл ресурсів завдяки тому, що перемикання між ними відбувається без блокування фізичних потоків процесора, а управління передається іншим задачам під час очікування. Відповідно до розроблених правил мережевого протоколу як сервер, так і обчислювальний елемент реалізують спеціалізовані методи для передачі інформації у визначених для них форматах.

Усю мережеву взаємодію в межах системи можна розділити на синхронну та асинхронну (неблокуючу). До синхронної взаємодії належить лише одна активність – організація обчислень підключених обчислювальних елементів. Виконання обчислень має ітеративний характер, перед початком якого серверний застосунок проходить етап планування. Зважаючи на автономність обчислювальних елементів, які можуть припинити участь в обчисленнях у будь-який момент, сервер виконує також асинхронні дії, зокрема постійне опитування обчислювальних елементів під час робочої ітерації. Це дозволяє більш ефективно планувати наступні ітерації. Для забезпечення такої взаємодії обчислювальний елемент також використовує кілька віртуальних потоків, зокрема потік для підтримки зв'язку із сервером та потік виконання обчислень. Рис. 1 демонструє процеси, що відбуваються під час робочої ітерації.

До асинхронних подій на серверному застосунку також належать: очікування та обробка нових підключених обчислювальних елементів, аналіз виконаних завдань і планування наступної робочої ітерації. Усі ці процеси спрямовані на досягнення основної мети – мінімізацію простою підключених обчислювальних елементів, що є ключовим завданням у подібних системах.

Автономна природа обчислювальних елементів не лише унеможливлює формування чіткого планування обчислень, а й накладає обмеження надійності отриманих результатів. Оскільки такі елементи можуть мати різне географічне розташування та не належать безпосередньо системі, не можливо перевірити ні їхню апаратну, ні програмну конфігурацію на предмет наявності зловмисного програмного забезпечення.

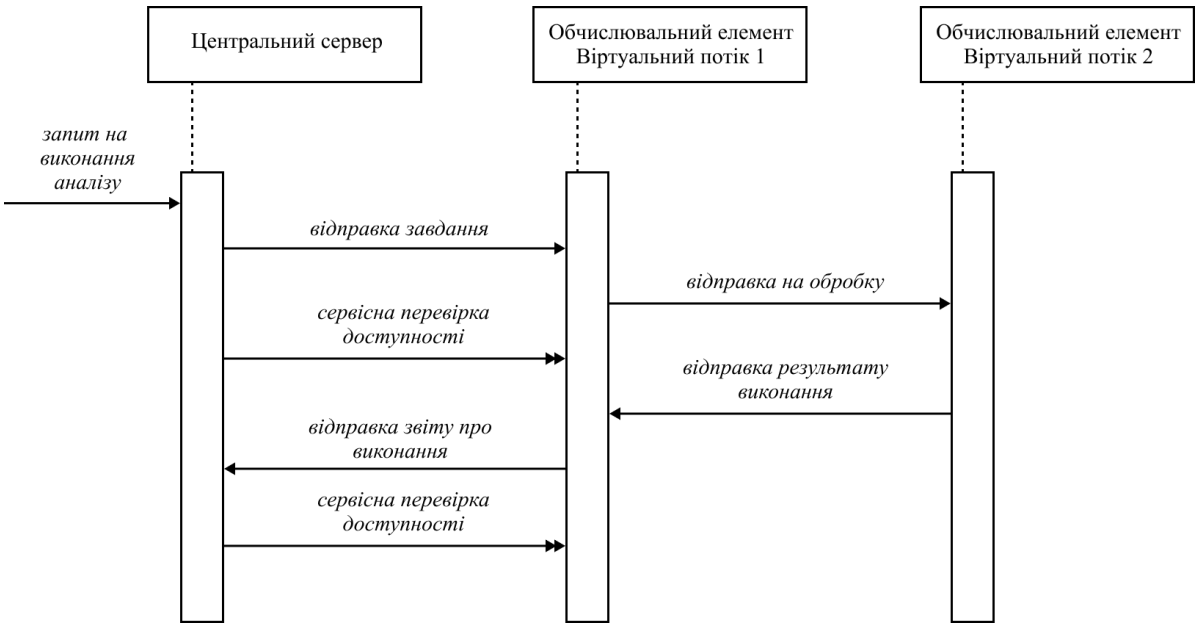

Рис. 1. Діаграма послідовності взаємодії сервера з обчислювальним елементом під час робочої ітерації

З огляду на це серверний застосунок для кожної підзадачі генеруватиме дублюючі завдання, які виконуватимуться іншими обчислювальними елементами. Це дозволить перевірити коректність отриманих результатів. Крім того, під час підключення нового обчислювального елемента до системи серверний застосунок надсилає йому тестове завдання. Це дає змогу оцінити його приблизну обчислювальну потужність і, відповідно, більш ефективно розподіляти завдання між елементами.

Метод розподілу завдань можна представити у вигляді таких послідовних кроків:

Отримання списку активних обчислювальних елементів у системі.

Перед початком нової ітерації серверний застосунок вибирає обчислювальні елементи, що мають відповідний статус. Зокрема, до цієї множини не включаються елементи, які ще не виконали тестове завдання. Таким чином, формується набір обчислювальних елементів, готових до участі в обчисленнях $CE_{active} = \{ce_{a_1}, ce_{a_2}, \dots, ce_{a_n}\}$. Окрім цього, для кожного елемента ініціалізується значення його поточного завантаження CE_{active}^{load}, яке на цьому етапі встановлюється на нуль

Аналіз черги завдань на серверному застосунку.

Коли нові задачі від користувачів потрапляють на опрацювання на серверний застосунок, він асинхронно виконує його оцінку складності виконання, тому на цьому етапі уже є сформована відповідна множина $T_{load} = \{t_{l_1}, t_{l_2}, \dots, t_{l_n}\}$. Також, серверний застосунок формує множину CE_{load}, що містить тільки ті елементи, які будуть приймати участь в наступній обчислювальній ітерації.

Визначення оптимального розподілу завдань між обчислювальними елементами.

На цьому етапі, серверний застосунок виконує пошук найкращого розподілу завдань, для цього він застосовує інформацію про складність кожної задачі, а також враховує поточний час завантаженості кожного обчислювального елементу. Враховуючи ці показники, він визначає кому розподілити задачу, керуючись (2):

$$ce_{index} = arg \min_{k \in \{1 \dots n\}} \left(ce_{active}^{load} + t_{ce_{index}}^{task}\right) | \forall ce_{active}^{load} \in T \quad (2)$$

Дубльовані задачі перед призначенням також додатково перевіряються на те, який саме обчислювальний елемент їх виконував. Таким чином, навіть якщо оптимальний розподіл передбачає призначення завдання такому самому елементу, що виконував його раніше, цей варіант буде відхилено. Це дозволяє забезпечити коректність виконання всіх поставлених перед системою завдань.

Ініціалізація робочої обчислювальної ітерації. Коли для всіх поставлених завдань визначено обчислювальні елементи, що їх виконуватимуть, серверний застосунок надсилає завдання всім активним учасникам і очікує їх виконання. Тривалість ітерації визначається адміністратором на власний розсуд і потребує компромісного рішення. Надто довгі ітерації можуть призвести до втрати частини результатів, якщо деякі обчислювальні елементи від'єднаються до завершення обчислень. Водночас занадто короткі ітерації можуть спричинити надмірні витрати часу на комунікацію.

Експерименти

Для оцінки ефективності запропонованого алгоритму розподілу завдань в обчислювальній системі було також реалізовано додаткові алгоритми планування, що дозволило провести порівняльний аналіз. Експериментальні дані отримано на основі роботи системи з п'ятьма обчислювальними елементами та 10 наборами задач, що містили до 100 завдань для розподілу. Результати експерименту представлено на рис. 2.

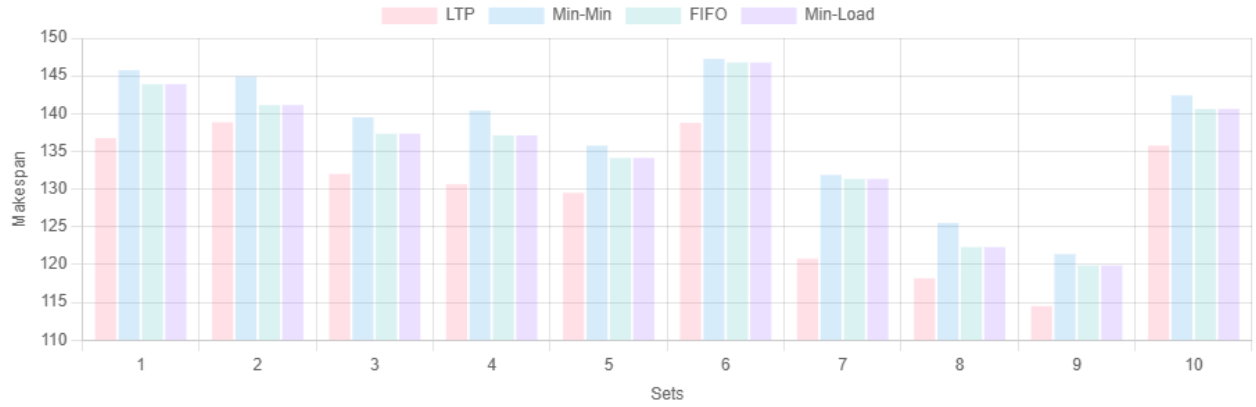

Рис. 2 – Порівняння загального часу виконання для алгоритмів розподілу задач

Графік ілюструє загальний час виконання всього набору завдань у системі при використанні визначеного алгоритму. Запропонований алгоритм продемонстрував ефективність на рівні 5-7% у порівнянні з іншими алгоритмами, такими як Min-Min, FIFO і Min-Load.

Висновки

У цій статті представлено аналіз використання технологій розподілених систем для забезпечення безпеки в IT. Розглянуто основні властивості грід-обчислювальних систем і запропоновано їхнє використання для виявлення зловмисної активності у програмах та файлах. Описано архітектуру систему, що передбачає використання обчислювальних елементів для розгортання ізольованого середовища виконання файлів з метою їхнього аналізу. Визначено основні характеристики серверного застосунку та обчислювального елемента, а також представлено мережевий протокол їхньої взаємодії. Запропонований метод демонструє ефективність у середньому на 5% вищу порівняно з іншими алгоритмами розподілу завдань, що застосовуються в подібних сценаріях. Подальші дослідження мають бути спрямовані на оцінку ефективності алгоритму в умовах випадкових відключень обчислювальних елементів.

Література

1. Yerina A. Statistical Indicators of Cybersecurity Development in the Context of Digital Transformation of Economy and Society / A. Yerina, I. Honchar, S. Zaiets // Science and Innovation. — 2021. — Vol. 17, No. 3. — P. 3–13.

2. Li Y. A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments / Y. Li, Q. Liu // Energy Reports. — 2021. — Vol. 7. — P. 8176–8186

3. SonicWall. 2024 Mid-Year Cyber Threat Report [Електронний ресурс]. — Режим доступу: https://www.sonicwall.com/threat-report (дата звернення: 11.02.2025)

4. Castillo A. Trojan Malware Detection using ANN, Naïve Bayes and SVM Machine Learning Algorithms / A. Castillo, A. B. Lineses, B. Go, R. Labanan, M. Octaviano // Proceedings of the 2nd International Conference in Information and Computing Research (iCORE), [December 2022] / IEEE. — P. 72–76.

5. Pilania S. Zeus: In-Depth Malware Analysis of Banking Trojan Malware / S. Pilania, R. S. Kunwar // Advanced Techniques and Applications of Cybersecurity and Forensics — Chapman and Hall/CRC, 2024.

6. Markowsky G. The Technique for Metamorphic Viruses' Detection Based on Its Obfuscation Features Analysis / G. Markowsky, O. Savenko, S. Lysenko, A. Nicheporuk // Proceedings of the ICTERI Workshops, [2018]. — P. 680–687.

7. Khoje M. A. Exploring Polymorphic Malware Analysis Techniques: A Comprehensive Survey / M. A. Khoje, A. Kaswan, J. Pawar // International Journal of Renewable Energy Exchange. — 2023. — Vol. 1, No. 1. — P. 136–142.

8. Badhwar R. Polymorphic and metamorphic malware / R. Badhwar // The CISO's Next Frontier: AI, Post-Quantum Cryptography and Advanced Security Paradigms. — Cham: Springer International Publishing, 2021. — P. 279–285.

9. Arora P. Malware Analysis Types & Techniques: A Survey / P. Arora, R. Gupta, N. Malik, A. Kumar // Proceedings of the 5th International Conference on Information Management & Machine Intelligence, [November 2023]. — P. 1–6.

10. Hakobyan R. Polymorphic Malware Analysis Model / R. Hakobyan, T. Jamgharyan // Computer Science and Information Technologies: Proceedings of the 14th International Conference, [Yerevan, Armenia], September 25–30 — Yerevan: National Academy of Sciences of Armenia, 2023. — P. 85–88.

11. Kashtalian A. The Technique for Metamorphic Viruses' Detection Based on Its Obfuscation Features Analysis / A. Kashtalian, S. Lysenko, O. Savenko, A. Nicheporuk, T. Sochor, V. Avsiyevych // Radioelectronic and Computer Systems. — 2024. — No. 1 (103). — P. 104–112.

12. Pomorova O. A technique for detection of bots which are using polymorphic code / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, A. Nicheporuk // Computer Networks: Proceedings of the 21st

International Conference, CN 2014, [Brunów, Poland, June 23–27, 2014] — Cham: Springer Publishing, 2014. — P. 265–276.

13. Aboaoja F. A. Malware detection issues, challenges, and future directions: A survey / F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-Rimy, T. A. E. Eisa, A. A. H. Elnour // Applied Sciences. — 2022. — Vol. 12, No. 17. — Article 8482.

14. Pomorova O. Metamorphic Viruses Detection Technique Based on the Modified Emulators / O. Pomorova, O. Savenko, S. Lysenko, A. Nicheporuk // Proceedings of the 12th International Conference on ICT in Education, Research, and Industrial Applications (ICTERI), [June 2016]. — P. 375–383.

15. Alqaraleh M. Enhanced Resource Discovery Algorithm for Efficient Grid Computing / M. Alqaraleh // *Proceedings of the 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, [June 2024] / IEEE. — P. 925–931.

16. Liu Q. Deep reinforcement learning for load-balancing aware network control in IoT edge systems / Q. Liu, T. Xia, L. Cheng, M. Van Eijk, T. Ozcelebi, Y. Mao // *IEEE Transactions on Parallel and Distributed Systems*. — 2021. — Vol. 33, No. 6. — P. 1491–1502.

17. Essien U. E. Cuckoo Sandbox and Process Monitor (Procmon) Performance Evaluation in Large-Scale Malware Detection and Analysis / U. E. Essien, S. I. Ele // Technology. — 2024. — Vol. 7, No. 4. — P. 8–26.

18. Vasani V. Comprehensive Analysis of Advanced Techniques and Vital Tools for Detecting Malware Intrusion / V. Vasani, A. K. Bairwa, S. Joshi, A. Pljonkin, M. Kaur, M. Amoon // Electronics. — 2023. — Vol. 12, No. 17. — Article 4299.

19. Hybrid Analysis [Електронний ресурс]. — Режим доступу: https://www.hybrid-analysis.com/faq (дата звернення: 11.02.2025) — Frequently Asked Questions

20. ANY.RUN: Features [Електронний ресурс]. — Режим доступу: https://any.run/features/ (дата звернення: 11.02.2025) — Analyze Malware and Phishing in a safe environment

21. Muñoz A. Joe Sandbox: The Best Cloud-Based Malware Analysis Service [Електронний ресурс] / A. Muñoz // *Medium*. — Режим доступу: https://alvaromartmunoz.medium.com/joe-sandbox-the-best-cloud-based-malware-analysis-service-cf7db35cecd8 (дата звернення: 11.02.2025)

22. Ali M. Agent-based vs agent-less sandbox for dynamic behavioral analysis / M. Ali, S. Shiaeles, M. Papadaki, B. V. Ghita // Proceedings of the 2018 Global Information Infrastructure and Networking Symposium (GIIS), [October 2018] / IEEE. — P. 1–5.

23. Pal D. Comparative Analysis of Cluster, Utility, Grid and Cloud Computing / D. Pal // *SSRN Electronic Journal.* — 2021. — P. 1–10.

24. Latif R. A novel trust management model for edge computing / R. Latif, M. U. Ahmed, S. Tahir, S. Latif, W. Iqbal, A. Ahmad // *Complex & Intelligent Systems*. — 2022. — Vol. 8, No. 1. — P. 469–487.

25. Регіда П. Метод виявлення зловмисної активності в інфікованих програмах / П. Регіда, О. Савенко // *Information Technology: Computer Science, Software Engineering and Cyber Security*. — 2024. — № 4. — С. 178–186.

References

1. Yerina A. Statistical Indicators of Cybersecurity Development in the Context of Digital Transformation of Economy and Society / A. Yerina, I. Honchar, S. Zaiets // Science and Innovation. — 2021. — Vol. 17, No. 3. — P. 3–13.

2. Li Y. A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments / Y. Li, Q. Liu // Energy Reports. — 2021. — Vol. 7. — P. 8176–8186

3. SonicWall. 2024 Mid-Year Cyber Threat Report [Electronic resource]. — Available at: https://www.sonicwall.com/threat-report (accessed: 11.02.2025)

4. Castillo A. Trojan Malware Detection using ANN, Naïve Bayes and SVM Machine Learning Algorithms / A. Castillo, A. B. Lineses, B. Go, R. Labanan, M. Octaviano // Proceedings of the 2nd International Conference in Information and Computing Research (iCORE), [December 2022] / IEEE. — P. 72–76.

5. Pilania S. Zeus: In-Depth Malware Analysis of Banking Trojan Malware / S. Pilania, R. S. Kunwar // Advanced Techniques and Applications of Cybersecurity and Forensics — Chapman and Hall/CRC, 2024.

6. Markowsky G. The Technique for Metamorphic Viruses' Detection Based on Its Obfuscation Features Analysis / G. Markowsky, O. Savenko, S. Lysenko, A. Nicheporuk // Proceedings of the ICTERI Workshops, [2018]. — P. 680–687.

7. Khoje M. A. Exploring Polymorphic Malware Analysis Techniques: A Comprehensive Survey / M. A. Khoje, A. Kaswan, J. Pawar // International Journal of Renewable Energy Exchange. — 2023. — Vol. 1, No. 1. — P. 136–142.

8. Badhwar R. Polymorphic and metamorphic malware / R. Badhwar // The CISO's Next Frontier: AI, Post-Quantum Cryptography and Advanced Security Paradigms. — Cham: Springer International Publishing, 2021. — P. 279–285.

9. Arora P. Malware Analysis Types & Techniques: A Survey / P. Arora, R. Gupta, N. Malik, A. Kumar // Proceedings of the 5th International Conference on Information Management & Machine Intelligence, [November 2023]. — P. 1–6.

10. Hakobyan R. Polymorphic Malware Analysis Model / R. Hakobyan, T. Jamgharyan // Computer Science and Information Technologies: Proceedings of the 14th International Conference, [Yerevan, Armenia], September 25–30 — Yerevan: National Academy of Sciences of Armenia, 2023. — P. 85–88.

11. Kashtalian A. The Technique for Metamorphic Viruses' Detection Based on Its Obfuscation Features Analysis / A. Kashtalian, S. Lysenko, O. Savenko, A. Nicheporuk, T. Sochor, V. Avsiyevych // Radioelectronic and Computer Systems. — 2024. — No. 1 (103). — P. 104–112.

12. Pomorova O. A technique for detection of bots which are using polymorphic code / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, A. Nicheporuk // Computer Networks: Proceedings of the 21st International Conference, CN 2014, [Brunów, Poland, June 23–27, 2014] — Cham: Springer Publishing, 2014. — P. 265–276.

13. Aboaoja F. A. Malware detection issues, challenges, and future directions: A survey / F. A. Aboaoja, A. Zainal, F. A.

Ghaleb, B. A. S. Al-Rimy, T. A. E. Eisa, A. A. H. Elnour // Applied Sciences. — 2022. — Vol. 12, No. 17. — Article 8482.

14. Pomorova O. Metamorphic Viruses Detection Technique Based on the Modified Emulators / O. Pomorova, O. Savenko, S. Lysenko, A. Nicheporuk // Proceedings of the 12th International Conference on ICT in Education, Research, and Industrial Applications (ICTERI), [June 2016]. — P. 375–383.

15. Alqaraleh M. Enhanced Resource Discovery Algorithm for Efficient Grid Computing / M. Alqaraleh // *Proceedings of the 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC),* [June 2024] / IEEE. — P. 925–931.

16. Liu Q. Deep reinforcement learning for load-balancing aware network control in IoT edge systems / Q. Liu, T. Xia, L. Cheng, M. Van Eijk, T. Ozcelebi, Y. Mao *// IEEE Transactions on Parallel and Distributed Systems.* — 2021. — Vol. 33, No. 6. — P. 1491–1502.

17. Essien U. E. Cuckoo Sandbox and Process Monitor (Procmon) Performance Evaluation in Large-Scale Malware Detection and Analysis / U. E. Essien, S. I. Ele // Technology. — 2024. — Vol. 7, No. 4. — P. 8–26.

18. Vasani V. Comprehensive Analysis of Advanced Techniques and Vital Tools for Detecting Malware Intrusion / V. Vasani, A. K. Bairwa, S. Joshi, A. Pljonkin, M. Kaur, M. Amoon // Electronics. — 2023. — Vol. 12, No. 17. — Article 4299.

19. Hybrid Analysis [Electronic resource]. — Available at: https://www.hybrid-analysis.com/faq (accessed: 11.02.2025) — Frequently Asked Questions

20. ANY.RUN: Features [Electronic resource]. — Available at: https://any.run/features/ (accessed: 11.02.2025) — Analyze Malware and Phishing in a safe environment

21. Muñoz A. Joe Sandbox: The Best Cloud-Based Malware Analysis Service [Electronic resource] / A. Muñoz // *Medium*. — Available at: https://alvaromartmunoz.medium.com/joe-sandbox-the-best-cloud-based-malware-analysis-service-cf7db35cecd8 (accessed: 11.02.2025)

22. Ali M. Agent-based vs agent-less sandbox for dynamic behavioral analysis / M. Ali, S. Shiaeles, M. Papadaki, B. V. Ghita // Proceedings of the 2018 Global Information Infrastructure and Networking Symposium (GIIS), [October 2018] / IEEE. — P. 1–5.

23. Pal D. Comparative Analysis of Cluster, Utility, Grid and Cloud Computing / D. Pal // *SSRN Electronic Journal.* — 2021. — P. 1–10.

24. Latif R. A novel trust management model for edge computing / R. Latif, M. U. Ahmed, S. Tahir, S. Latif, W. Iqbal, A. Ahmad // *Complex & Intelligent Systems*. — 2022. — Vol. 8, No. 1. — P. 469–487.

25. Rehida P. Metod vyiavlennia zlovmysnoi aktyvnosti v infikovanykh prohramakh [Method for detecting malicious activity in infected programs] / P. Rehida, O. Savenko // Information Technology: Computer Science, Software Engineering and Cyber Security. — 2024. — No. 4. — P. 178–186.