УДК 004.49:004.75
DOI https://doi.org/10.32782/IT/2024-4-21

Павло РЕГІДА
старший викладач кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет, вул. Інститутська 11, м. Хмельницький, Україна, 29016
***ORCID:** 0000-0002-6591-7069*
***Scopus Author ID:** 57202317133*

Олег САВЕНКО
доктор технічних наук, професор, професор кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет, вул. Інститутська 11, м. Хмельницький, Україна, 29016
***ORCID:** 0000-0002-4104-745X*
***Scopus Author ID:** 54421023400*

Бібліографічний опис статті: Регіда, П., Савенко, О. (2024). Метод виявлення зловмисної активності в інфікованих програмах. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 4, 178–186, doi: https://doi.org/10.32782/IT/2024-4-21

МЕТОД ВИЯВЛЕННЯ ЗЛОВМИСНОЇ АКТИВНОСТІ В ІНФІКОВАНИХ ПРОГРАМАХ

Розробники зловмисного програмного забезпечення (ЗПЗ) застосовують різноманітні техніки маскування, що дозволяють уникнути виявленню під час їх виконання на хості користувача. Серед ключових класів таких технік визначають зокрема: анти-емуляційні та поліморфні, через їх можливість протидіяти гібридному методу виявлення, який часто застосовується в сучасних АПЗ. Саме тому, дослідження методів маскування, їх комбінування та загальний вплив на хост корчувача дозволить сформувати метод їх виявлення. В зв'язку із цим, буде доречно застосувати технології емуляції, пісочниці та розподілених систем.

***Метою статті** є виявлення зловмисної поведінки в інфікованих програмах із техніками уникнення виявлення на основі аналізу зміни поведінки їх виконання в модифікованих ізольованих середовищах.*

***Методологія** полягає у застосуванні наукових методів: синтез, аналіз та порівняння. В роботі представлений аналіз частини сучасних методів уникнення виявлення, що застосовуються в інфікованих програмах. Розглянуто засіб виконання інфікованої програми, та архітектуру системи для організації розподіленого виявлення зловмисної поведінки. Представлено алгоритм формування поведінки виконання програми в ізольованому середовищі.*

***Наукова новизна** полягає у формуванні методу виявлення зловмисного прояву у інфікованих програмах, що застосовують захисні техніки протидії емуляції та поліморфізму. Приділено увагу стратегії виконання таких інфікованих програм в різних середовищах, що позначається на їх поведінці виконання. Представлені результати досліджень, що демонструють ефективність запропонованого методу в рамках досліджених технік уникнення виявлення.*

***Висновки.** Запропонований метод дозволяє покращити виявлення інфікованих програм, які використовують поліморфні техніки перетворення, для уникнення виявлення в ситуації коли емульоване середовище нею було визначено. Представлено архітектуру системи, що забезпечує множину модифікованих ізольованих середовищ для проведення аналізу виконання програм.*

***Ключові слова:** виявлення вірусів, емуляція, пісочниця, маскувальні техніки.*

Pavlo REHIDA
Senior Lecturer at the Department of Computer Engineering and Information Systems, Khmelnytskyi National University, 11, Instytuts'ka Str., Khmelnytskyi, Ukraine 29016, pavlo.rehida@gmail.com
***ORCID:** 0000-0002-6591-7069*
***Scopus Author ID:** 57202317133*

Oleg SAVENKO
Doctor of Technical Sciences, Professor, Professor at the Department of Computer Engineering and Information Systems, Khmelnytskyi National University, 11, Instytuts'ka Str., Khmelnytskyi, Ukraine, 29016, savenko_oleg_st@ukr.net
***ORCID:** 0000-0002-4104-745X*
***Scopus Author ID:** 54421023400*

To site this article: Rehida, P., Savenko, O. (2024). Metod vyiavlennia zlovmysnoi aktyvnosti v infikovanykh prohramakh. [Method for detecting malicious activity in infected programs]. *Information Technology: Computer Science, Software Engineering and Cyber Security,* 4, 178–186, doi: https://doi.org/10.32782/IT/2024-4-21

METHOD FOR DETECTING MALICIOUS ACTIVITY IN INFECTED PROGRAMS

Developers of malwares employ various avoid techniques to evade detection during execution on user host. Key classes of such techniques include anti-emulation and polymorphic techniques, due to their ability to counteract hybrid detection methods commonly used in modern antivirus programs. Therefore, studying avoid techniques, their combinations, and their impact on the host will help in developing detection methods. In this context, it is appropriate to use emulation technologies, sandboxes, and distributing systems.

***The purpose** is to detect malicious activity in infected programs that use evasion techniques by analysing changes in their execution behaviour in modified isolated environments.*

***The methodology** involves the use of scientific methods: synthesis, analysis, and comparison. The paper presents an analysis of modern evasion techniques in infected programs. It discusses the execution tool for infected programs and the system architecture for organizing distributed detection of malicious activity. An algorithm to form program execution behaviour in an isolated environment is also presented.*

***The scientific novelty** lies in developing a method for detecting, malicious activity in infected programs that use anti-emulation and polymorphic techniques. Special attention is given to the strategy of executing such programs in different environments, which impacts their execution behaviour. The presented research results demonstrate the effectiveness of proposed method within the scope of studied evasion techniques.*

***Conclusions.** The proposed method improves the detection of infected programs that use polymorphic transformation techniques to evade detection, particularly when the emulated environment has been identified. The architecture of the presented system provides a set of modified isolated environments for analysing program execution.*

***Key words:** malware detection, emulation, sandbox, avoid techniques.*

Актуальність проблеми. Для вирішення поставної задачі виявлення наявності зловмисного прояву в інфікованій програмі запропоновано використати розподілений підхід із використанням різних модифікованих ізольованих середовищ. Основною відмінністю середовищ є застосування досліджених технік анти-емуляції в якості параметрів їх модифікації. Сучасне зловмисне програмне забезпечення використовує велику кількість технік уникнення від виявлення антивірусними програмними засобами. Тому, пропонується провести аналіз виконання програм в різних умовах, що дозволить виявити відмінність у виконанні, і тим самим підтвердити наявність зловмисного прояву.

Аналіз останніх досліджень і публікацій. Розробники сучасних екземплярів зловмисного програмного забезпечення досліджують техніки їх виявлення підходами, які використовуються в антивірусних програмних засобах. Основною метою таких досліджень, є формування нових стратегій для приховання зловмисного прояву. Аналіз звітів наукових лабораторій та наукових публікацій тематика яких є дослідження ЗПЗ, а також їх методів впливу на хост користувача показують, що як і кількість так і методи впливу із кожним роком все збільшуються (Razaulla, 2023; Savenko 2021). Метою створення таких ЗПЗ може бути і модифікація існуючого програмного забезпечення, яке застосовується отримання як фінансової вигоди, так і для отримання несанкціонованого доступу до хосту користувач (Savenko, 2020), та навіть отримання доступу до елементів кіберфізичних систем (Lysenko, 2022). Для того щоб вирішити цю проблему, фахівці кібербезпеки почали розробляти та застосовувати антивірусні програмні засоби (АПЗ), як основний засіб захисту хосту користувача. Враховуючи динаміку появи нових ЗПЗ та методів їх впливу, сучасні АПЗ постійно вдосконалюються шляхом аналізу нових загроз від ЗПЗ та впровадженням стратегій їх протидії.

В більшості сучасних настільних та мобільних операційних системах по замовчуванню використовуються вбудоване АПЗ (Numminen, 2023), яке використовує велику кількість підходів для виявлення ЗПЗ. В літературі, такі підходи узагальнюють у три основні методи виявлення: статичний (Omar, 2022) динамічний (Leon, 2021) та гібридний (Yunus, 2020). Окрім вбудованих методів, широко застосовуються і сторонні рішення, які відрізняються розширеним функціоналом, що представлені додатковими методами виявлення.

Методи виявлення сучасних АПЗ базуються на використанні гібридного методу виявлення. Динамічний аналіз здійснюється в ізольованому середовищі з емуляцією апаратних компонентів хоста, що дозволяє безпечно виконувати аналіз програм на наявність ЗПЗ та його

поведінки. Крім того, АПЗ активно використовують бази вже відомих загроз при застосуванні статичного аналізу. Щоб протидіяти таким методам виявлення, розробники ЗПЗ наділяють їх властивостями, що дозволяють підвищити шанс уникнення. Такі властивості представлені у вигляді технік, які базуються на досліджених методах виявлення. В літературі такі техніки, розділяються на категорії за методом аналізу або впливу на хост користувача. (Faruki, 2023)

Серед основних технік уникнення виявлення визначають техніки протидії емуляції, через те, що сучасні засоби виявлення базуються на використанні ізольованих середовищ із емуляцією. Саме тому, для ЗПЗ важливо виявити наявність емуляції в середовищі виконання, адже його наявність із високою імовірністю означає використання АПЗ. Такі техніки спрямовані для пошуку невідповідності у характеристиках виконуваного середовища у порівнянні із реальним середовищем користувача, які базуються на аналізі: швидкості виконання інструкцій процесора (D'Elia, 2020), ідентифікації апаратних засобів (Apostolopoulos, 2021), пошук артефактів емуляції (Nappa, 2021).

Ще однією важливою категорією таких технік визначають як поліморфні. За допомогою такого підходу, в окремих випадках, ЗПЗ може змінювати код виконання інфікованої програми. Такий підхід дозволяє оминути статичний метод (Nicheporuk, 2020) виявлення, який базується на порівнянні із уже відомими ЗПЗ, які знаходяться у відповідній базі АПЗ. Це відбувається завдяки використанню відповідних технік, які можуть змінювати код виконання програми, при цьому зберігаючи її функціональні властивості (Gorment, 2023). Зазвичай, для такої модифікації використовуються декілька технік одночасно, до яких входять: використання додаткового коду (Gibert, 2022), перепризначення регістру, заміни інструкцій (Han, 2022) та розділення коду.

Для використання ізольованого середовища з метою аналізу програм на наявність зловмисної активності, використовують технологію емуляції апаратних засобів хоста (Fedák, 2022). Це також дозволяє отримати більше можливостей в плані аналізу процесу виконання програми (Liu, 2022) та сформувати поведінку виконання програми. Дослідження поведінки, в свою чергу, дозволить визначити наявність зловмисної поведінки (Arabo, 2020) у програмі, що була проаналізована.

Аналіз виконання програми стає можливим із використанням за допомогою зчитування вмісту основних регістрів процесору (Schaik, 2021) за допомогою використання програмних переривань (Amit, 2015).

Успішне застосування розподілених систем у наукових та практичних завданнях (Peng, 2020) сприяло їх широкому розвитку. Особливу категорію таких систем становлять такі, що базуються на грід-технологіях (Hamdan, 2020). Хоча організація цих систем вимагає вирішення додаткових завдань, пов'язаних з гарантуванням безпеки (Kong, 2022; Rehida 2023) та коректності їхньої роботи, вони знаходять широке застосування у сучасних дослідженнях та розробках.

Мета дослідження. Метою дослідження є розробка методу виявлення інфікованих програм, що використовують техніки уникнення виявлення на основі аналізу виконання програми в модифікованих ізольованих середовищах.

Виклад основного матеріалу дослідження. Розробники ЗПЗ застосовують для виконання зловмисної дії використовують різні підходи, і одним і ключових напрямків є інфікування файлів. Інфіковане програмне забезпечення отримує додаткові функціональні властивості, як наприклад: виконати зловмисну дію. Розглянуті техніки уникнення виявлення, також стають частиною інфікованих програм. Тому, якщо представити набір технік уникнення емуляції як $M_{ae} = \{ae_1, ae_2 \ldots ae_n\}$, а техніки поліморфізму як $M_{pt} = \{pt_1, pt_1 \ldots pt_m,\}$. При цьому, кількість технік, які використовуються в рамках одного вірусу не може налічувати усі їх варіації, так як така суттєва модифікація оригінального програмного забезпечення (ПЗ) буде викликати підозру навіть зі сторони користувача. Нехай оригінальне ПЗ та ЗПЗ це *sw* та *mw* відповідно. Тоді інфіковану програму можна представити у наступному вигляді:

$$Isw_i = \{sw_i, mw_i, M_{ae_i}, M_{pt_i}\} \quad (1)$$

Так як мета будь-якого вірусу виконати, який є частиною інфікованої програми виконати свою зловмисну дію, він може змінювати власну стратегію виконання на основі результатів виконання цих стратегій. Тому, у випадку коли емуляція виявлена, інфіковане програмне може приховати свою зловмисну поведінку. І для того щоб виявити наявність зловмисної поведінки, пропонується дослідити поведінку виконання програми у різних модифікованих середовищах. Розглянуті раніше техніки модифікації коду з метою обходу виявлення, часто застосовують саме цю зміну на рівні інструкцій, що подаються на виконання в Центральний процесор (ЦП).

Для пошуку такої відмінності було розроблено базовий емулятор, який може опрацьовувати декілька десятків інструкцій низького

рівня різного типу, такі як: арифметичні, логічні, із плаваючої комою, строкові, потоків керування, роботи із даними та системні. З точки зору виконання таких інструкцій, компоненти ЦП можна розділити на активні та пасивні. Активними компонентами будемо визначати такі, стан яких змінюється при виконанні інструкцій. При виконання набору інструкцій, які собою представляють певну програму, поступово змінюється вміст регістрів. Тому, визначимо стан ЦП як множину набору значень в поточний момент виконання програми. І враховуючи, що в сучасних процесорах здебільшого використовуються наступні групи регістрів: основні регістри – $R_b = \{r_{RAX}, r_{RBX}, r_{RCX}, r_{RDX}\}$, регістри вказівники – $R_p = \{r_{RBP}, r_{RSP}\}$, строкові регістри – $R_s = \{r_{RSI}, r_{RDI}\}$ а також загальні регістри – $R_r = \{r_8 \dots r_{15}\}$, то поточний стан ЦП можемо представити як:

$$CPU_{state} = \{R_b, R_p, R_s, R_r\} \quad (2)$$

Тоді, враховуючи що кожна програма містить велику кількість інструкції, то її поведінку можна представити у вигляді набору таких станів. Дослідження станів дозволить виявити власне чи відбулась зміна виконання програми у різних модифікованих середовищах. А, для модифікації середовища використовуються знання про існуючі техніки виявлення емуляції. В таких середовищах, вони будуть використовуватись в якості приманок, а цю складову частину можемо представити як $EE_{ae} = \{ae_1, ae_2 \dots ae_n\}$. Тому, слід очікувати, що інфікована програма буде виконуватись по різному в середовищах які мають і не мають визначені модифікації. Тому при використанні деякої кількості таких середовищ із різними параметрами, формування поведінки для кожного із них буде виглядати наступним чином

$$B_{Isw_i} = \begin{cases} EE_1\big|_{EE_{ae_1}} \xrightarrow{Isw_i} B_1 \,\big|\, EE_{ae_1} \subset M_{ae_i} \\ EE_2\big|_{EE_{ae_2}} \xrightarrow{Isw_i} B_2 \,\big|\, EE_{ae_2} \cap M_{ae_i} = \varnothing \end{cases} \quad (3)$$

В результаті аналізу завдяки такому підходу, буде сформована множина поведінок, і для подальшого виявлення потрібно буде порівняти усі отримані поведінки.

Для забезпечення такого підходу, буде доцільно сформувати розподілену систему. Запропонована система має бути централізованою, а обчислювальні елементи (ОЕ) автономними та гетерогенними. Це дозволить об'єднати велику кількість ОЕ для вирішення поставленого завдання. Основним завданням ОЕ буде проводити аналіз надісланих зразків потенційних інфікованих програм, та формувати поведінку його виконання. Саме тому, можемо визначити ряд основних функцій CE_f, до яких будуть входити: $ce_{f_{gt}}$ – отримання завдання від ЦС, $ce_{f_{ie}}$ – ініціалізація модифікованого емульованого середовища, $ce_{f_{pe}}$ – запуск на виконання отриманого зразка ІП, $ce_{f_{fs}}$ – формування стану ЦП в визначений момент часу, $ce_{f_{cb}}$ – формування поведінки програми на основі накопичених станів ЦП, $ce_{f_{sr}}$ – відправка результату аналізу на центральний сервер. Тоді, список функцій такого спеціалізованого ПЗ, буде виглядати наступним чином:

$$CE_f = \{ce_{f_{gt}}, ce_{f_{ie}}, ce_{f_{pe}}, ce_{f_{fs}}, ce_{f_{cb}}, ce_{f_{sr}}\}. \quad (4)$$

Основними завданнями які визначені перед центральним сервером є: надавати інтерфейсу для користувачів, що хочуть проаналізувати підозрілі програми, аналізувати стан системи на кількість ОЕ, формувати завдання для активних ОЕ, проводи аналіз отриманих результатів. Тому, можемо визначити набір основних функцій центрального серверу CS_f, а саме: $cs_{f_{ut}}$ – отримання зразка інфікованої програми від користувача, $cs_{f_{fa}}$ – формування активних ОЕ в поточний момент часу, $cs_{f_{tf}}$ – формування завдання на основі активних елементів, $cs_{f_{ar}}$ – збір результатів аналізу програми, $cs_{f_{dr}}$ – визначення наявності зловмисного прояву, $cs_{f_{rb}}$ – сповіщення користувача про результат розподіленого аналізу програми. Отже, список функцій центрального серверу буде так:

$$CS_f = \{cs_{f_{ut}}, cs_{f_{fa}}, cs_{f_{tf}}, cs_{f_{ar}}, cs_{f_{dr}}, cs_{f_{rb}}\}. \quad (5)$$

Запропонована система зображена на рисунку 1, який також представляє які основні модулі приймають участь для забезпечення проведення розподіленого аналізу.

Метод розподіленого виявлення інфікованих програм на основі аналізу поведінки виконання. Представимо запропонований метод більш детально, у вигляді кроків.

1. *Отримання системою ІП на аналіз.* Для користувача, при роботі із системою, передбачений відповідний інтерфейс, який дозволяє завантажувати підозрілий файл *IS* на аналіз. Завантажені файли у систему записуються у чергу CS_q, звідти вибираються коли система вільна. В разі чого, формується множина:

$$CS_q = \{Isw_1, Isw_2, \dots, Isw_n\}. \quad (6)$$

2. *Формування списку активних ОЕ та генерація параметрів для ізольованих середовищ.* Так як розглянута система працює із набором автономних ОЕ, центральний сервер

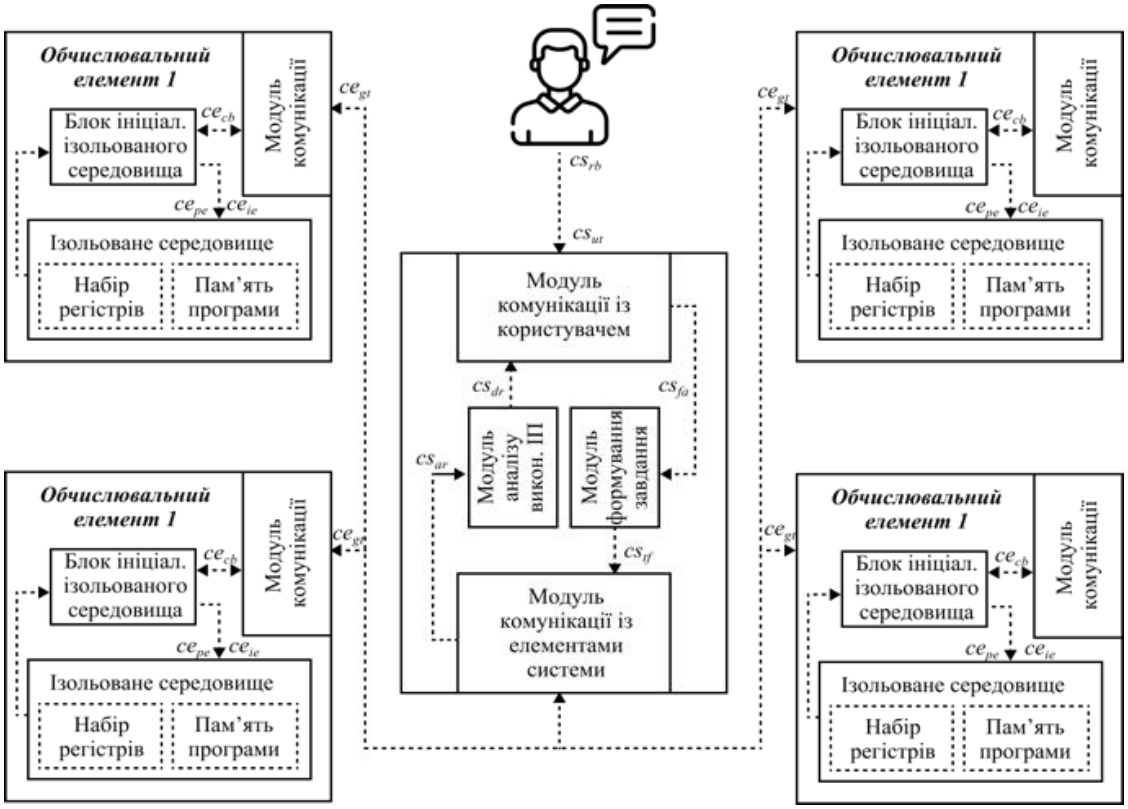

Рис. 1. Архітектура системи розподіленого виявлення зловмисного прояву

перевіряє ті хто долучені в систему, і відповідно до цього формує набір параметрів для ізольованих середовищ. Таким чином, для розподіленого аналізу ІП, використовується наступний набір параметрів:

$$T_{lsw\,i} = \left\{ Isw_i, EE^1_{ae_1} \ldots EE^1_{ae_k}, EE^2_{ae_1} \ldots EE^2_{ae_l}, \ldots, EE^n_{ae_1} \ldots EE^n_{ae_m} \right\}. \quad (7)$$

де, n – кількість активних ОЕ, на яких можна розгорнути ізольоване середовище, а k,l,m – це кількість пасток яка застосовується для кожного середовища відповідно.

3. *Генерація ізольованих середовищ на ОЕ*. Кожен ОЕ від центрального сервера отримає наступний множину, для виконання поставленого завдання $T^n_{IS_i} = \left\{ Isw_i, EE^n : EE^n = \left\{ EE^n_{ae_1}, \ldots, EE^n_{ae_m} \right\} \right\}$. На основі отриманого набору ОЕ формує модифіковане ізольоване середовище, та ініціює емулятор для виконання програми. На цьому етапі, система працює із n набором середовищ, що будуть аналізувати ІП що знаходиться у черзі під номером i , що буде мати наступний вигляд.

$$Isw_i = \left\{ EE^1, EE^2, \ldots, EE^n \right\}. \quad (8)$$

4. *Аналізу програми під час виконання її на ОЕ.* Для проведення аналізу виконання програми запропоновано використати програмні переривання, для зчитування даних про стан регістрів. Враховуючи те, що інформація про зміну стану процесора буде накопичуватись, то визначений стан (2) варто подати наступним чином:

$$CPU_{state} = \left\{ \vec{R}_b, \vec{R}_p, \vec{R}_s, \vec{R}_r \right\} \quad (9)$$

У виразі (9) усі групи регістрів представлені у вигляді векторів, так як кожен вектор містить інформацію про зміну стану відповідного до його регістру. Але такий запис не є зручним для проведення порівняння, тому пропонується виконати наступні перетворення, для формування єдиного значення, що буде характеризувати поведінку в цілому:

$$B = FH\left(\sum_{i=0}^{n} FH\left(\sum_{j=0}^{m} r_i \right) \right) \quad (10)$$

де, i – кількість виконаних програмних переривань, j – кількість регістрів ЦП, та FH функція хешування.

5. *Аналіз отриманих результаті центральним сервером.* Після того, як усі ОЕ повернуть результати виконання програм в різних модифікованих ізольованих середовищах, центральний сервер формує набір поведінок $SB_{lsw_i} = \left\{ B^1_{lsw_i}, B^2_{lsw_i}, \ldots, B^n_{lsw_i} \right\}$. Оскільки сучасні віруси завдяки своїм методам захисту можуть змінювати свою поведінку, центральний сервер буде керуватись правилом (11) для виявлення зловмисного прояву.

$$\neg\left(\forall x, y \in \left\{B^1_{Isw_j}, B^2_{Isw_j}, \ldots, B^n_{Isw_j}\right\}, x = y\right) \Rightarrow Isw_i \text{ is malware} \quad (11)$$

Після чого, про результат аналізу буде сповіщений користувач. На цьому етапі усі ОЕ знищують свої ізольовані середовища, та переходять у режим очікування. Запропонований метод дозволяє підвищити виявлення для ІП які застосовують техніки виявлення емуляції та поліморфні техніки, а також стратегії що базуються на поєднанні їх особливостей. Результативність використання такого методу залежить від кількості досліджених технік уникнення та правильному формуванню наборів параметрів для ізольованих середовищ в розподіленій системі з автономними ОЕ. Запропонований метод графічно представлено на рисунку 2.

Для проведення експериментів, було згенеровано 4 моделі ІП, які використовують визначену кількість технік виявлення емуляції середовища, а також 2 набори модифікованих ізольованих середовищ по 8 у кожному. Таблиця 1 демонструє визначені техніки ІП, а також модифікації середовищ. Таблиця 2 показує зафіксовану кількість різних поведінок в результаті проведення аналізу.

Висновки. У дослідженні запропоновано модель системи для проведення аналізу поведінки виконання ІП в ізольованому середовищі. Даний підхід передбачає формування модифікованих середовищ, на основі знань про техніки виявлення емульованого середовища в якості пасток. Це дозволить створити умови, при яких ІП буде змушена використати одну із технік поліморфізму, що дозволить уникнути виявленню. Для проведення експерименту було створено 4 моделі ІП, які реагують на модифікації в ізольованих середовищах згідно дослідженнями в проаналізованих наукових публікаціях. Кожна модель була проаналізована в двох наборах середовищ із відмінним

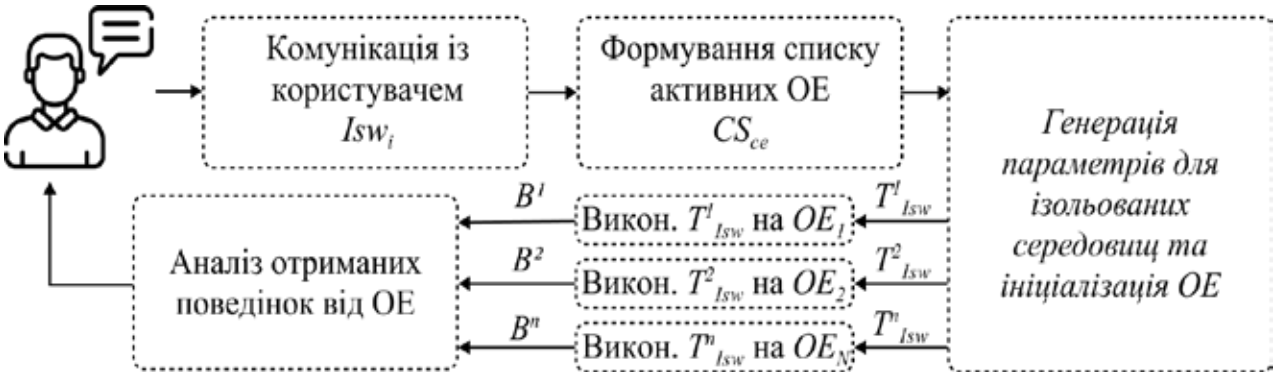

Рис. 2. Графічне представлення методу виявлення зловмисного прояву

Таблиця 1

Вхідні дані для проведення експерименту з дослідження поведінки ІП

ІП	*Набір м. середовищ 1 (GEE_1)*		*Набір м. середовищ 2 (GEE_2)*	
$Isw_1(ae_1, ae_2)$	$EE^1_1(ae_1, ae_2)$	$EE^1_5(ae_1, ae_5)$	$EE^2_1(ae_4, ae_5)$	$EE^2_5(ae_5)$
$Isw_2(ae_3)$	$EE^1_2(ae_4)$	$EE^1_6(ae_5)$	$EE^2_2(ae_1)$	$EE^2_6(ae_1, ae_6)$
$Isw_3(ae_3, ae_5)$	$EE^1_3(ae_4, ae_5)$	$EE^1_7(ae_2)$	$EE^2_3(ae_5, ae_6)$	$EE^2_7(ae_2, ae_6)$
$Isw_4(ae_2, ae_6)$	$EE^1_4(ae_1, ae_4)$	$EE^1_8(ae_2, ae_6)$	$EE^2_4(ae_2, ae_4)$	$EE^2_8(ae_4)$

Таблиця 2

Результати експерименту виявлення зловмисної поведінки

ІП	*Поведінки у GEE_1*	*Поведінки у GEE_2*	*Результат*
$Isw_1(ae_1, ae_2)$	5 x $B^1_{Isw_1}$, 3 x $B^2_{Isw_1}$	4 x $B^1_{Isw_1}$, 4 x $B^2_{Isw_1}$	*виявлено*
$Isw_2(ae_3)$	8 x $B^1_{Isw_2}$	8 x $B^1_{Isw_2}$	*не виявлено*
$Isw_3(ae_3, ae_5)$	3 x $B^1_{Isw_3}$, 5 x $B^2_{Isw_3}$	3 x $B^1_{Isw_3}$, 5 x $B^2_{Isw_3}$	*виявлено*
$Isw_4(ae_2, ae_6)$	3 x $B^1_{Isw_4}$, 5 x $B^2_{Isw_4}$	4 x $B^1_{Isw_4}$, 4 x $B^2_{Isw_4}$	*виявлено*

набором пасток, за допомогою централізованої системи. Відмінності поведінки зафіксовані для 1-ї, 3-ї та 4-ї моделей. Щодо 2-ї моделі, так як жодне з середовищ не використовувало модифікацію згідно із її технікою виявлення емуляції, поведінка залишилась однакова. Кількість досліджених технік уникнення виявлення, що застосовується ІП, та їх використання для модифікації середовища виконання, напряму підвищує шанс на виявлення нових. Тому, предметом подальших досліджень є вивчення нових технік захисту ІП від виявлення, їх комбінація в рамках одного екземпляру ІП, та стратегії їх застосування.

ЛІТЕРАТУРА:

1. Razaulla S., Fachkha C., Markarian C., Gawanmeh A., Mansoor W., Fung B. C., Assi C. The age of ransomware: A survey on the evolution, taxonomy, and research directions. *IEEE Access*. 2023. Vol. 11. P. 40698–40723. DOI: 0.1109/ACCESS.2023.3268535

2. Savenko B., Lysenko S., Bobrovnikova K., Savenko O., Markowsky G. Detection DNS tunneling botnets. *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, 22 September, 2021. P. 64-69. DOI: 10.1109/IDAACS53288.2021.9661022

3. Savenko O., Sachenko A., Lysenko S., Markowsky G.,Vasylkiv N. Botnet detection approach based on the distributed systems. *International Journal of Computing.* 2020. Vol. 19. № 2. P. 190–198. DOI: 10.47839/ijc.19.2.1761

4. Lysenko S., Bobrovnikova K., Kharchenko V., Savenko O. IoT multi-vector cyberattack detection based on machine learning algorithms: traffic features analysis, experiments, and efficiency. *Algorithms.* 2022. Vol 15. № 7. P. 239. DOI: 10.3390/a15070239

5. Numminen A. Windows technical hardening against the most prevalent threats: Master Thesis : 2023, 61 p.

6. Omar M. Static Analysis of Malware. *Defending Cyber Systems through Reverse Engineering of Criminal Malware.* 2022. P. 9–17. DOI: 10.1007/978-3-031-11626-1_2

7. Leon, R. S., Kiperberg M., Leon Zabag A. A., Zaidenberg N. J. Hypervisor-assisted dynamic malware analysis. *Cybersecurity.* 2021. Vol. 4. P. 1–14. DOI: 10.1186/s42400-021-00083-9

8. Yunus Y. K. B. M., Ngah S. B. Review of hybrid analysis technique for malware detection. *The 6th International Conference on Software Engineering & Computer Systems*, Vol. 769, 25–27 September, 2019, Pahang, Malaysia. P. 012075. DOI: 10.1088/1757-899X/769/1/012075

9. Faruki P., Bhan R., Jain V., Bhatia S., El Madhoun N., Pamula R. A survey and evaluation of android-based malware evasion techniques and detection frameworks. *Information.* 2023. Vol. 14. № 7. P.374. DOI: 10.3390/info14070374

10. D'Elia D. C., Coppa E., Palmaro F., Cavallaro L. On the dissection of evasive malware. *IEEE Transactions on Information Forensics and Security*. 2020. Vol. 15. P. 2750–2765. DOI: 10.1109/TIFS.2020.2976559

11. Apostolopoulos T., Katos V., Choo K. K. R., Patsakis C. Resurrecting anti-virtualization and anti-debugging: Unhooking your hooks. *Future Generation Computer Systems.* 2021. Vol. 116. P. 393–405. DOI: 10.1016/j.future.2020.11.004

12. Nappa A., Papadopoulos P., Varvello M., Gomez D.A., Tapiador J., Lanzi A. Pow-how: An enduring timing side-channel to evade online malware sandboxes. *26th European Symposium on Research in Computer Security,* 4-8 October, 2021, Darmstadt, Germany. P. 86–109. DOI: 10.1007/978-3-030-88418-5_5

13. Nicheporuk A., Savenko O., Nicheporuk A., Nicheporuk Y. An Android Malware Detection Method Based on CNN Mixed-Data Model. *16th International Conference on ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer,* Vol. 2732, October 06-10, 2020, Kharkiv, Ukraine. P. 198–213.

14. Gorment N. Z., Selamat A., Krejcar O. Obfuscated malware detection: impacts on detection methods. *Asian Conference on Intelligent Information and Database Systems*, Vol. 1863, 2023. P. 55–66. DOI: 10.1007/978-3-031-42430-4_5

15. Gibert D., Fredrikson M., Mateu C., Planes J., Le Q. Enhancing the insertion of NOP instructions to obfuscate malware via deep reinforcement learning. *Computers & Security.* 2022. Vol. 113. P. 102543. DOI: 10.1016/j.cose.2021.102543

16. Han S. H., Lee D. Kernel-based real-time file access monitoring structure for detecting malware activity. *Electronics.* 2022. Vol. 11, № 12, 1871. DOI: 10.3390/electronics11121871

17. Fedák A., Štulrajter J. Evasion of antivirus with the help of packers. *Science & Military Journal.* 2022. Vol. 17, № 1. P 14–22. DOI: 10.52651/sam.a.2022.1.14-22

18. Liu S., Feng P., Wang S., Sun K., Cao J. Enhancing malware analysis sandboxes with emulated user behavior. *Computers & Security.* 2022. Vol. 115, 102613. DOI: 10.1016/j.cose.2022.102613

19. Arabo A., Dijoux R., Poulain T., Chevalier G. Detecting ransomware using process behavior analysis. *Procedia Computer Science.* 2020. Vol. 168. P 289–296. DOI: 10.1016/j.procs.2020.02.249

20. Van Schaik S., Minkin M., Kwong A., Genkin D., Yarom Y. CacheOut: Leaking data on Intel CPUs via cache evictions. *IEEE Symposium on Security and Privacy (SP).* 24-27 May, 2021, San Francisco, CA, USA. P. 339–354. DOI: 10.1109/SP40001.2021.00064

21. Amit N., Tsafrir D., Schuster A., Ayoub A., Shlomo, E. Virtual CPU validation. *Proceedings of the 25th Symposium on Operating Systems Principles.* 04 October, 2015. P. 311–327. DOI: 10.1145/2815400.2815420

22. Peng P., Soljanin E., Whiting P. Diversity vs. parallelism in distributed computing with redundancy. *IEEE International Symposium on Information Theory (ISIT).* 21-26 June, 2020, Los Angeles, CA, USA. P. 257–262. DOI: 10.1109/ISIT44484.2020.9174030

23. Hamdan S., Ayyash M., Almajali, S. Edge-computing architectures for internet of things applications: A survey. *Sensors.* 2020. Vol. 20, № 22, P. 6441. DOI: 10.3390/s20226441

24. Kong X., Wu Y., Wang H., Xia F. Edge computing for internet of everything: A survey. *IEEE Internet of Things Journal.* 2022. Vol 9, № 23, P. 23472–23485. DOI: 10.1109/JIOT.2022.3200431

25. Rehida P., Sochor T., Martynyuk V., Tarasova O., Orlenko V. A distributed malware detection model based on sandbox technology. *Intelligent Information Technologies & Systems of Information Security (IntelITSIS)*, Vol. 3373, 22-24 March, 2023, Khmelnytskyi, Ukraine. P. 475–485.

REFERENCES:

1. Razaulla, S., Fachkha, C., Markarian, C., Gawanmeh, A., Mansoor, W., Fung, B. C., & Assi, C. (2023). The age of ransomware: A survey on the evolution, taxonomy, and research directions. *IEEE* Access, 11, 40698–40723. https://doi.org/10.1109/ACCESS.2023.3268535

2. Savenko, B., Lysenko, S., Bobrovnikova, K., Savenko, O., & Markowsky, G. (2021, September). Detection DNS tunneling botnets. *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, 1, 64–69, IEEE. https://doi.org/10.1109/IDAACS53288.2021.9661022

3. Savenko, O., Sachenko, A., Lysenko, S., Markowsky, G., & Vasylkiv, N. (2020). Botnet detection approach based on the distributed systems. *International Journal of Computing*, *19*(2), 190–198. https://doi.org/10.47839/ijc.19.2.1761

4. Lysenko, S., Bobrovnikova, K., Kharchenko, V., & Savenko, O. (2022). IoT multi-vector cyberattack detection based on machine learning algorithms: traffic features analysis, experiments, and efficiency. *Algorithms*, *15*(7), 239. https://doi.org/10.3390/a15070239

5. Numminen, A. (2023). *Windows technical hardening against the most prevalent threats* [Master thesis, University of Jyväskylä]. JYX Digital Repository. Retrieved from: https://jyx.jyu.fi/handle/123456789/87270

6. Omar, M. (2022). Static Analysis of Malware. *Defending Cyber Systems through Reverse Engineering of Criminal Malware* (pp. 9–17). Springer, Cham. https://doi.org/10.1007/978-3-031-11626-1_2

7. Leon, R. S., Kiperberg, M., Leon Zabag, A. A., & Zaidenberg, N. J. (2021). Hypervisor-assisted dynamic malware analysis. *Cybersecurity*, 4, 1–14. https://doi.org/10.1186/s42400-021-00083-9

8. Yunus, Y. K. B. M., & Ngah, S. B. (2020, February). Review of hybrid analysis technique for malware detection. *IOP conference series: materials science and engineering* 769(1), 012075). IOP Publishing. https://doi.org/10.1088/1757-899X/769/1/012075

9. Faruki, P., Bhan, R., Jain, V., Bhatia, S., El Madhoun, N., & Pamula, R. (2023). A survey and evaluation of android-based malware evasion techniques and detection frameworks. *Information*, 14(7), 374. https://doi.org/10.3390/info14070374

10. D'Elia, D. C., Coppa, E., Palmaro, F., & Cavallaro, L. (2020). On the dissection of evasive malware. *IEEE Transactions on Information Forensics and Security*, *15*, 2750–2765. https://doi.org/10.1109/TIFS.2020.2976559

11. Apostolopoulos, T., Katos, V., Choo, K. K. R., & Patsakis, C. (2021). Resurrecting anti-virtualization and anti-debugging: Unhooking your hooks. *Future Generation Computer Systems*, 116, 393–405. https://doi.org/10.1016/j.future.2020.11.004

12. Nappa, A., Papadopoulos, P., Varvello, M., Gomez, D. A., Tapiador, J., & Lanzi, A. (2021). Pow-how: An enduring timing side-channel to evade online malware sandboxes. *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security,* 26, 86–109. Springer International Publishing. https://doi.org/10.1007/978-3-030-88418-5_5

13. Nicheporuk, A., Savenko, O., Nicheporuk, A., & Nicheporuk, Y. (2020, October). An Android Malware Detection Method Based on CNN Mixed-Data Model. *16th International Conference on ICT in Education,*

Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer (ICTERI), 2732, pp. 198–213. Retrieved from: https://ceur-ws.org/Vol-2732/20200198.pdf

14. Gorment, N. Z., Selamat, A., & Krejcar, O. (2023, July). Obfuscated malware detection: impacts on detection methods. *Asian Conference on Intelligent Information and Database Systems*, 55-66. Springer, Cham. https://doi.org/10.1007/978-3-031-42430-4_5

15. Gibert, D., Fredrikson, M., Mateu, C., Planes, J., & Le, Q. (2022). Enhancing the insertion of NOP instructions to obfuscate malware via deep reinforcement learning. *Computers & Security*, 113, 102543. https://doi.org/10.1016/j.cose.2021.102543

16. Han, S. H., & Lee, D. (2022). Kernel-based real-time file access monitoring structure for detecting malware activity. *Electronics*, 11(12), 1871. https://doi.org/10.3390/electronics11121871

17. Fedák, A., & Štulrajter, J. (2022). Evasion of antivirus with the help of packers. *Science & Military Journal*, *17*(1), 14–22. https://doi.org/10.52651/sam.a.2022.1.14-22

18. Liu, S., Feng, P., Wang, S., Sun, K., & Cao, J. (2022). Enhancing malware analysis sandboxes with emulated user behavior. *Computers & Security*, 115, 102613. https://doi.org/10.1016/j.cose.2022.102613

19. Arabo, A., Dijoux, R., Poulain, T., & Chevalier, G. (2020). Detecting ransomware using process behavior analysis. *Procedia Computer Science*, 168, 289–296. https://doi.org/10.1016/j.procs.2020.02.249

20. Van Schaik, S., Minkin, M., Kwong, A., Genkin, D., & Yarom, Y. (2021, May). CacheOut: Leaking data on Intel CPUs via cache evictions. *2021 IEEE Symposium on Security and Privacy (SP),* 339–354. IEEE. https://doi.org/10.1109/SP40001.2021.00064

21. Amit, N., Tsafrir, D., Schuster, A., Ayoub, A., & Shlomo, E. (2015, October). Virtual CPU validation. *Proceedings of the 25th Symposium on Operating Systems Principles*, 311–327. https://doi.org/10.1145/2815400.2815420

22. Peng, P., Soljanin, E., & Whiting, P. (2020, June). Diversity vs. parallelism in distributed computing with redundancy. *2020 IEEE International Symposium on Information Theory*, 257–262. IEEE. https://doi.org/10.1109/ISIT44484.2020.9174030

23. Hamdan, S., Ayyash, M., & Almajali, S. (2020). Edge-computing architectures for internet of things applications: A survey. *Sensors*, 20(22), 6441. https://doi.org/10.3390/s20226441

24. Kong, X., Wu, Y., Wang, H., & Xia, F. (2022). Edge computing for internet of everything: A survey. *IEEE Internet of Things Journal*, 9(23), 23472–23485. https://doi.org/10.1109/JIOT.2022.3200431

25. Rehida, P., Sochor, T., Martynyuk, V., Tarasova, O., & Orlenko, V. (2023). A distributed malware detection model based on sandbox technology. 4th International Workshop on Intelligent Information Technologies & Systems of Information Security *IntelITSIS*, 3373, 475–485. Retrieved from: https://ceur-ws.org/Vol-3373/paper32.pdf