УДК 004.092

ФОРМУВАННЯ СИГНАТУРИ ПОВЕДІНКИ ПРОГРАМИ НА ОСНОВІ ТРАСУВАННЯ API ВИКЛИКІВ

О. С. Савенко, А. О. Нічепорук, А. А. Нічепорук, Ю. О. Нічепорук
Хмельницький національний університет

Анотація. *Запропоновано метод для формування сигнатури вірусної програми на основі трасування API викликів. Розроблена сигнатура дозволяє здійснити віднесення вірусної програми не тільки до одного із класів вірусів, але й дозволяє визначати його модифікацію. Розроблений метод для проведення експериментальних досліджень реалізовано в розподіленій багаторівневій системі виявлення зловмисного програмного забезпечення в локальних мережах.*

Ключові слова: *вірусна програма, трасування API викликів, хі-квадрат тест, поведінка*

Вступ

Сучасні темпи інформатизації суспільства перевищують темпи розвитку всіх інших галузей. Масове застосування персональних комп'ютерів виявилося пов'язаним з поширенням комп'ютерних вірусів. На сьогоднішній день вірусами є складні програми, які здатні приховувати власне функціонування, викрадати персональні дані користувачів, самовідтворюватись та блокувати роботу як окремих комп'ютерних систем так і цілих мереж. При цьому тенденція поширення шкідливого програмного забезпечення, а особливо вірусів, набуває загрозливого характеру. Так згідно із даними антивірусної компанії McAfee [1], у 2017 кількість нових зразків шкідливого програмного коду склала близько 57,6 мільйонів, що майже вдвічі більше ніж у 2016 році.

Для виявлення шкідливого програмного забезпечення існує ряд як комерційних, так і теоретичних методів. В основі більшості сучасних антивірусних засобів для виявлення шкідливого програмного забезпечення є сигнатурний аналіз [2]. Він передбачає пошук ділянок програмного коду або символьних рядків, що дозволяє однозначно визначити наявність шкідливого програмного забезпечення. Така технологія дозволяє з високою достовірністю та низьким рівнем хибних спрацювань виявляти різні види шкідливого програмного забезпечення. Проте, застосування сигнатурного аналізу для виявлення вірусів, що застосовують технології заплутування коду є не ефективним. Головним чином це пов'язано з відсутністю сталої складової, яку можна виокремити в якості сигнатури.

Постановка задачі дослідження

Використання у вірусах технології заплутування програмного коду унеможливлює виокремлення сталої частини коду, аналіз якої дозволив би прийняти рішення про можливе інфікування. Проте, це стає можливим при використанні в якості основи сигнатури API викликів функцій, тобто набору готових класів, процедур, функцій, структур і констант, що надаються додатком або операційною системою для використання у зовнішніх програмних продуктах.

Процес виконання програм, зокрема вірусів, супроводжується використанням ними API викликів. Наприклад, вірусна програма для пошуку виконуваних файлів, які повинні бути інфіковані, як правило, використовує наступну послідовність API викликів: FindFirstFileA, FindNextFileA і FindClose, що розміщені в бібліотеці KERNEL32.DLL. Таким чином зазначена послідовність API викликів може бути використана для побудови сигнатури вірусної програми. В загальному, можна відзначити, що використання API функцій в якості сигнатури, дозволяє виокремити сталу семантичну (поведінкову) складову, в той час як синтаксична складова буде різною.

Сформуємо вимоги, яким має відповідати сигнатура вірусної програми на основі трасування API викликів наступним чином:

- аналіз сигнатури має дозволяти визначати не тільки клас вірусних програм (сімейство), а й модифікацію вірусу (наприклад у вірусу virut є модифікації ce, a, b, тощо);

- буди компактною (мати не великий розмір). Відомо, що зі збільшенням розміру сигнатури збільшується розмір бази даних сигнатур та час аналізу цієї сигнатури, і відповідно, час виявлення;

- алгоритми виявлення, що засновані на аналізі вірусної сигнатури повинні володіти прийнятною часовою та обчислювальною складністю;

- алгоритми виявлення, що використовують запропоновану сигнатуру мають володіти високою достовірністю виявлення та низьким рівнем хибних спрацювань.

Огляд попередніх досліджень

На сьогоднішній день проблемі виявлення вірусних програм на основі відстеження API викликів приділяється значна увага [3-10, 15, 16].

У роботі [3] представлено підхід до виявлення шкідливого програмного забезпечення на основі динамічного аналізу API викликів. В основі запропонованого методу закладено визначення загального шаблону, що представлений послідовностями API викликів, на основі використання алгоритму множинного вирівнювання послідовностей (MSA). Процес виявлення передбачає співставлення шаблону підозрілої програми із базою поведінок з залученням алгоритму пошуку найбільшої спільної послідовності (LCS). Окрім того, з метою підвищення ефективності виявлення, до складу бази поведінок входить послідовність критичних дій, зіставлення з якими, однозначно визначає приналежність підозрілої програми до вірусу.

Автори роботи [4] запропонували процес формування вірусної сигнатури шляхом побудови поведінки програми. Поведінка програми представляється поведінковим автоматом, що формується на основі використання графу потоку виконання API викликів. Для здійснення процесу виявлення здійснюється відображення поведінкового автомату у множину k-грам. Пошук з відомими поведінковими шаблонами здійснюється на основі метрик подібності.

У роботі [5] автори використали метод реверс інженерії для отримання системних викликів з таблиці імпорту адрес (IAT) формату PE-EXE. Після отримання списку системних викликів здійснюється обрахунок частоти їх використання у довірених додатках та вірусних програмах. Для проведення класифікації залучено методи машинного навчання. Проте, у випадку аналізу запакованого шкідливого програмного забезпечення системні виклики з таблиці IAT будуть належати запакованому рівню, а не фактичному, що не відповідає реальній поведінці програми.

В роботі [6] запропоновано автоматичну систему дизасемблювання та отримання списку API викликів, що використовуються в додатках. Отримана множина API викликів відображається у векторний простір на основі залучення n-грамного аналізу. Для прийняття рішення про присутність шкідливого програного забезпечення використовується метод опорних векторів.

Інший підхід до формування сигнатури на основі трасування API викликів передбачає формування сигнатури для цілого вірусного класу [7]. Спочатку здійснюється дизасемблювання множини вірусних програм, що належать до одного сімейства, та отримують список API викликів. Далі, на основі хі-квадрат тесту здійснюється визначення значення приналежності кожного екземпляру, що використовувався для формування сигнатури. Після отримання усередненого значення приналежності до кожного класу вірусних програм, здійснюється співставлення поведінки досліджуваної програми з кожним класом на основі емпіричних правил.

У роботі [8] запропоновано використання ієрархії сценаріїв на основі API викликів для представлення знань про поведінку вірусних програм. Запропонований підхід передбачає використання моделі ієрархій сценаріїв шляхом залучення понять ролі, лісу ієрархій, а також зв'язуючої їх функції підтримки, що надало можливість за допомогою методів, заснованих на машинному навчанні, поряд з експертом, формувати сценарії із екземплярів вірусних програм в автоматизованому режимі.

Зазначені вище підходи до формування сигнатури на основі трасування API викликів дозволяють здійснити процес виявлення вірусних програм, проте, основним їх недоліком є відсутність протидії застосуванню фейкових API викликів (API виклики, що навмисно вводяться до вірусної програми з метою заплутування поведінки), що може ускладнити процес пошуку найбільшої спільної послідовності викликів, і відповідно, призвести до зменшення рівня виявлення.

Формування сигнатури поведінки програми на основі трасування API викликів

Для здійснення процесу виявлення вірусної програми розроблено систему, що складається з трьох основних складових: сигнатура вірусної програми, база сигнатур та метод виявлення, що передбачає зіставлення сигнатури вірусу з базою сигнатур.

При побудові сигнатури вірусу замість використання всіх API викликів, що здійснює вірусна програма, будуть розглядатися лише критичні API функції [9,10]. Критичні API виклики містять усі виклики API, які можуть призвести до порушення безпеки, зміни усталеної поведінки роботи системи або виклики, що використовуються для комунікації (модифікація значення системного реєстру, файлового вводу-виводу,

API доступу до мережевих ресурсів (WinSock), тощо). Слід зазначити, що в процесі створення сигнатури вірусної програми не розглядаються API виклики, які можна додати або вилучати з вірусної програми без зміни її шкідливої поведінки (наприклад MessageBox, printf, malloc тощо).

Сигнатура поведінки програми на основі трасування API викликів може бути представлена у вигляді сукупності двох складових (рис.1): частота виклику та характер взаємодії критичних API викликів. Аналіз першої складової дозволяє визначити розподіл критичних API викликів за групами шкідливої активності та відображає кількісну складову сигнатури. Друга складова сигнатури передбачає відображення у векторний простір характеру взаємодії критичних API функцій вірусної програми та описує їх взаємозв'язок. Аналіз другої складової сигнатури надає можливість розмежувати вірусні програми від корисних додатків не тільки за наявністю критичних API викликів, але й за їх взаємодією між собою.

Для опису характеру взаємодії критичних API викликів представимо вірусну програму у вигляді орієнтованого графа:

$$G_V = \langle V, E \rangle, \quad (1)$$

де V – множина вершин, що представляють групу критичних API функцій, а E – множина переходів між групами критичних API функцій, послідовність яких описує поведінку вірусної програми.

Для формального подання сигнатури вірусної програми представимо її у вигляді кортежу:

$$S = \langle A, F, \langle D, d_G, n_E \rangle \rangle, \quad (2)$$

де A – множина API викликів, що здійснює вірусна програма класу C_i, в процесі власного функціонування; F – множина частот виклику критичних API функцій; D – вектор степенів вершин графу G_V; d_G – діаметр графу G_V; n_E – кількість ребер графу G_V.

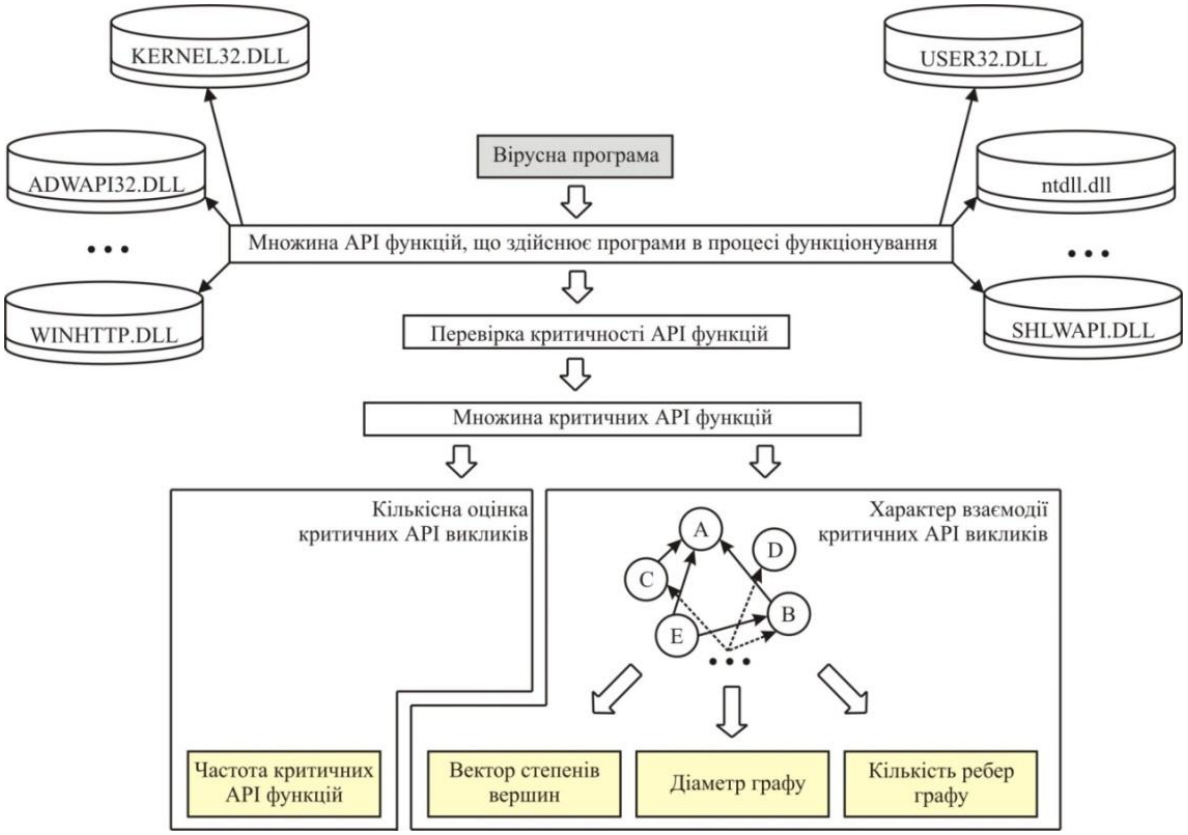

Рис. 1. Процес формування сигнатури поведінки вірусної програми на основі трасування API викликів

При формуванні вірусної сигнатури на основі трасування API викликів спільним для обох етапів є категоризація API викликів за класами. В процесі аналізу та дослідження множини вірусних програм було отримано 1624 API функції, які можна віднести до класу критичних API функцій. Для створення сигнатури вірусних програм зазначені API функції було розділено на 26 класів [11]. У таблиці 1 представлені приклади класів API функцій та їх опис. Наприклад, функції DeleteFiles та CreateDirectory визначаються як клас B, тобто функції для роботи з файлами та директоріями. У випадку якщо послідовність API викликів представляється функціями CallNextHookEx, isDebuggerPresent та CreateProcess, то в якості складової сигнатури

отримаємо наступну послідовність "AFH". Зазначене представлення API викликів дозволяє компактно зберігати поведінку програми, яка представлена API викликами. Окрім того, групування API функцій за класами критичних дій надає можливість представити у вигляді одного позначення множину функцій, які є подібні за функціональними властивостями (наприклад, CreateProcessAsUser та CreateProcess є подібними за виконуваними функціями) та можуть використовуватись різними екземплярами, що належать одному вірусному сімейству. Також слід відзначити, що в процесі формування сигнатури та категоризації критичних API викликів не враховуються вхідні параметри та результат виконання відповідної API функції.

Таблиця 1

Категоризація API функції по класам, їх опис та приклади

Клас API	Опис	Приклад	Кількість API
Class A	Функції перехоплення	CallNextHookEx, SetWindowsHookEx	12
Class B	Робота з файлами та директоріями	DeleteFiles, CreateDirectory, CopyFile	242
Class C	Модифікація системного реєстру	RegCreateKey, RegDeleteValue	48
Class D	Синхронізація	CreateMutex,CreateMutexEx	213
…	…	…	…
Class Z	Функції керування пристроями	DeviceControl, DvdLauncher	24

Аналіз множини частот виклику критичних API викликів дозволяє сформувати параметр приналежності до вірусного класу, що визначає зв'язок між вірусною програмою та одним із класів вірусних програм за кількістю критичних API викликів. Це є достатньою умовою для віднесення підозрілої програми до одного із класів вірусних програм чи корисних додатків. Проте, аналіз даного параметру не несе інформацію про характер взаємодії між критичними API викликами, і відповідно, не можливо віднести підозрілу програму до конкретної модифікації вірусу, а лише до цілого класу.

Тому друга складова сигнатури вірусної програми покликана відображати характер взаємодії критичних API функцій вірусної програми та описувати взаємозв'язок між ними, що дозволить здійснити розмежування вірусних програм в середині класу.

З цією метою вірусну програму можна представити у вигляді орієнтованого графа (1). Для відображення даного графу у сигнатурі вірусної програми представимо його у вигляді множини степенів вершин D, тобто кожний елемент цієї множини визначає число інцидентних ребер для відповідної вершини. Наприклад, за множиною ступенів вершин графу, що складається з послідовності {3,3,2,2,1,1} можна побудувати наступні графи (рис.2):

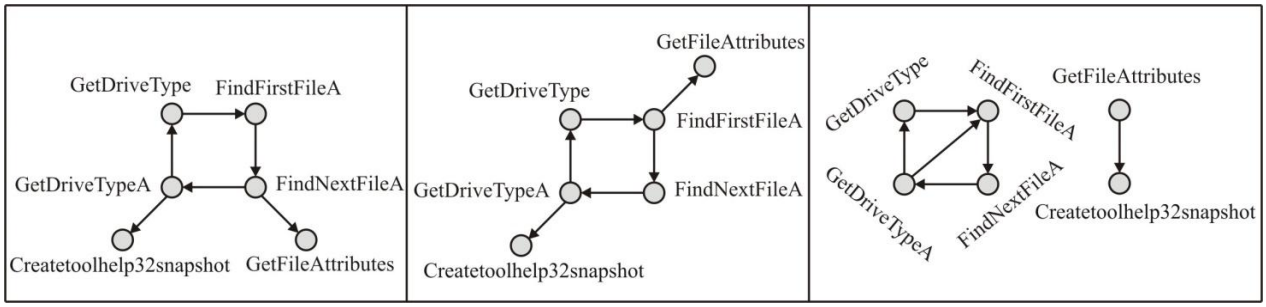

Рис. 2. Зображення графів зі степенями вершин {3,3,2,2,1,1}

Зазначені на рис. 2 прості приклади графів з однаковою степеню вершин характеризуються наявністю постійної складової (з'єднання у вигляді квадрата). Подібні закономірності притаманні і для вірусних програм.

Окрім множини степенів вершин для розмежування вірусних програм в середині класу виокремимо ще дві ознаки: діаметр графу та кількість ребер. Діаметр графу визначатиме максимальну послідовність виклику критичних API функцій, в той час як кількість ребер – загальну кількість виконуваних вірусною програмою дій.

Визначення параметру приналежності до вірусного класу

Однією із складових запропонованої сигнатури є множина частот виклику критичних API викликів. На основі елементів цієї множини здій-

снюється визначення параметру приналежності до вірусного класу. Він визначає ступінь належності екземпляра вірусу до одного із класів вірусних програм та дозволяє оцінити зв'язок між вірусними програмами в межах одного класу. Таким чином, база сигнатур, окрім частот виклику критичних API функцій, повинна містити параметр приналежності до вірусного класу.

Процес визначення параметру приналежності до вірусного класу заснований на відмінності між кількістю API викликів, які здійснюють вірусні програми та довірені додатки в процесі власного функціонування. Тому розмежування між класами вірусних програм та довірених додатків можливе за їх поведінкою, тобто послідовністю критичних API викликів. Розглянемо детальніше процес визначення параметру приналежності до вірусного класу.

З метою побудови поведінки класу вірусних програм $C_i = \{c_i^1, c_i^2, ..., c_i^x\}$ на основі частот виклику критичних API функцій представимо поведінку екземпляра цього класу у вигляді кортежу (c_i^j – екземпляр класу C_i, $j = \overline{1, x}$, де x – кількість екземплярів вірусних програм, що визначають поведінку класу C_i):

$$c_i^j = \langle f_1, f_2, ..., f_{26} \rangle, \quad (3)$$

де $f_1, f_2, ..., f_{26}$ – частоти критичних API функцій.

Згрупуємо всі значення частот виклику критичних API функцій c_i^j ($\forall c_i^j \in C_i$) та представимо їх у вигляді матриці R_{C_i}:

$$R_{C_i} = \begin{bmatrix} c_i^1 = \langle f_1, f_2, ..., f_{26} \rangle \\ c_i^2 = \langle f_1, f_2, ..., f_{26} \rangle \\ ... \\ c_i^x = \langle f_1, f_2, ..., f_{26} \rangle \end{bmatrix}, \quad (4)$$

На основі сформованої матриці R_{C_i} визначимо поведінку вірусних програм класу C_i як множину середніх значень викликів кожного класу критичних API функцій:

$$S_{C_i} = \langle F_1, F_2, ..., F_{26} \rangle, \quad (5)$$

де кожне значення F_i визначається як:

$$F_i = \frac{1}{x} \sum_{j=0}^{x} f_j, \quad (6)$$

де j – клас критичних API функцій.

На основі отриманої поведінки вірусного класу S_{C_i} здійснюється визначення параметру приналежності до вірусного класу C_i з використанням хі-квадрат тесту. Хі-квадрат тест визначає максимальну ймовірність тесту статистичної значущості, яка вимірює різницю між пропорціями в двох незалежних зразках.

Представимо сигнатуру S_{C_i} для класу вірусних програм C_i та кожну з поведінок екземплярів c_i^j у вигляді таблиці спряження (таблиця 2).

Таблиця 2

Вигляд таблиці спряження для сигнатури S_{C_i} для класу вірусних програм C_i та поведінки екземпляру c_i^j

Клас	Class A	Class B	…	Class Z	Всього
Сигнатура S_{C_i}					
Екземпляр c_i^j					
Всього					

Тоді, для отримання параметру приналежності до вірусного класу C_i за допомогою хі-квадрат тесту визначимо різницю між пропорціями в сигнатурі вірусного класу S_{C_i} та кожної з поведінок екземплярів c_i^j з поправкою на неперервність (з поправкою Йетса) наступним чином:

$$\chi_j^2 = \sum_{l=1}^{26} \frac{(|c_{i,l}^j - S_{C_{i,l}}| - 0.5)^2}{S_{C_{i,l}}}, \quad (7)$$

де l – відповідний клас критичних API викликів.

В результаті виконання даного етапу отримаємо множину пар значень (χ_i^2, c_i^j).

Наступний етап методу передбачає визначення усередненого значення параметру приналежності до вірусного класу C_i. З цією метою середнє значення визначатимемо наступним чином:

$$\mu_{C_i} = \frac{1}{x} \sum_{i=1}^{x} \chi_i^2. \quad (8)$$

Таким чином параметр μ_{C_i} визначає ступінь належності екземпляра c_i^j до вірусного класу та дозволяє оцінити наскільки сильним є зв'язок вірусних програм в межах одного класу C_i.

Виявлення вірусної програми, що представлена сигнатурою поведінки на основі трасування API викликів

Після створення сигнатури поведінки програми на основі трасування API викликів та ви-

значення параметру приналежності для кожного класу вірусних програм та довірених додатків розглянемо послідовність кроків, необхідних для виявлення вірусної програми, що представляється заданою сигнатурою.

Нехай для кожного класу вірусних програм та корисних додатків визначено параметр приналежності μ_{C_i} до відповідного класу та сигнатура підозрілої програми *S*, аналіз якої дозволить зробити висновок про присутність вірусу визначеного типу.

Перший крок методу виявлення передбачає визначення приналежності підозрілої програми до одного із класу вірусних програм або корисних додатків (рис.3). З цією метою за допомогою хі-квадрат тесту (7) визначається різниця між пропорціями частоти критичних API викликів підозрілої програми (перша частина сигнатури *S*) та частотою критичних API викликів кожного класу (5).

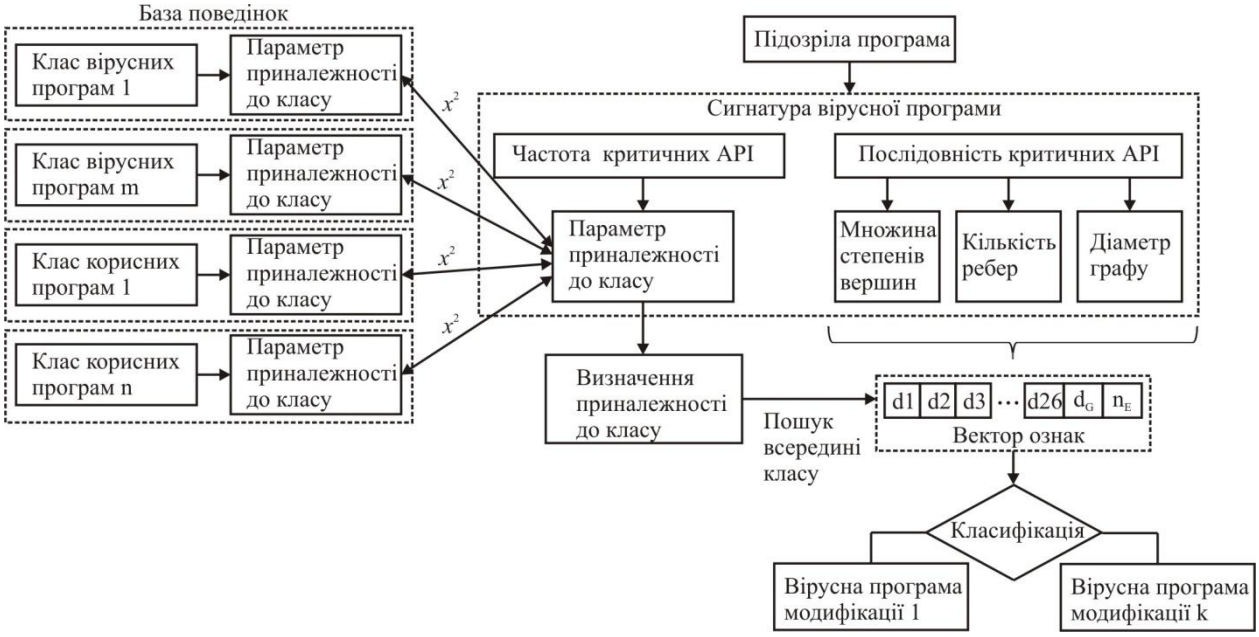

Рис. 3. Процес виявлення вірусних програм з використанням запропонованої сигнатури

В результаті даного етапу буде отримано множину значень параметру приналежності підозрілої програми до кожного із класів. Тоді клас, що найкраще відповідає заданій сигнатурі, визначається виходячи з виконання наступної умови:

$$\min(|\ \mu_{S_j} - \mu_{C_i}\ |) \qquad (9)$$

де μ_{S_j} – значення параметру приналежності підозрілої програми до кожного із *j* класів. В результаті буде визначено клас C_i, який за частотою критичних API викликів відповідає частоті викликів критичних API функцій підозрілої програми.

Наступний етап методу передбачає пошук вірусної сигнатури в межах класу C_i. Опишемо другу складову сигнатури (2) у вигляді вектора ознак *V*:

$$V = \langle D, d_G, n_E \rangle \qquad (10)$$

Зазначений вектор складається з 28 числових ознак, в якому 26 ознак визначають степені вершин графа (кожна вершина графа визначається класом критичних API викликів), а останні дві – діаметр графу та кількість ребер.

Після формування вектора ознак здійснюється його класифікація засобами машинного навчання, що дозволяє віднести досліджувану підозрілу програму до однієї з модифікацій вірусу.

Експериментальні дослідження

На основі запропонованої сигнатури було проведено експериментальні дослідження для оцінки ефективності виявлення вірусних програм.

Для проведення експерименту було використано 280 зразків вірусних програм, що отримані з ресурсу VX Heavens [12]. Всі вірусні програми належать до вірусних сімейств: Delf, Bifrose та MyDoom різних модифікацій (таблиця 2). Окрім вірусних програм було залучено 54 корисних додатки, що є виконуваними файлами операційної системи Windows (mspaint, bfsvc, тощо).

На першому етапі було сформовано базу поведінок для трьох класів Delf, Bifrose, MyDoom. З цією метою для кожного класу було визначено параметр приналежності з використанням хі-квадрат тесту (3-8).

На другому етапі було здійснено визначення рівня ефективності виявлення вірусних програм в межах класу.

Для побудови сигнатури вірусної програми розроблено програмне забезпечення, що використовує дані, отримані з використанням програми API Monitor [14]. Детальніше розроблене програмне забезпечення представлено в [15].

Для проведення експерименту всю множину вірусних програм, що представлені у вигляді розробленої сигнатури, було розділено на дві частини: навчальна та тестова вибірка. Навчальна вибірка складалась з 67 вірусних програм та 20 корисних додактів. Решта зразків шкідливого та корисного програмного забезпечення була використана для тестування. Схему проведення процесу класифікації наведено на рис. 4.

В якості алгоритму прийняття рішення було обрано J48 та Naive Bayes, що входять до вільно розповсюджуваного програмного забезпечення машинного навчання Weka [13].

Для визначення загального рівня ефективності виявлення було використано формулу, що заснована на використанні статистичних показників True Positive, True Negative, False Positive та False Negative:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \qquad (11)$$

де, TP – кількість вірно виявлених вірусних програм, FN – кількість хибно класифікованих вірусних програм, TN – кількість вірно ідентифікованих корисних програм, FP – кількість корисних програм неправильно класифікованих як вірусні програми.

Для визначення оцінки бінарної класифікації із застосуванням алгоритмів J48 та Naive Bayes було проведено ROC аналіз із побудовою кривих помилок для вірусних програм Delf, Bifrose та MyDoom (усереднено для всіх варіантів), що наведено на рис. 5. За результатами проведених експериментів кращі результати класифікації були відмічені при залучені алгоритму J48 для всіх трьох типів вірусних програм.

Загальна ефективність виявлення вірусних програм на основі запропонованої сигнатури наведено в таблиці 3. Найкращий показник виявлення було зафіксовано для вірусних програм класу MyDoom (93%). Слід відзначити, що у випадку помилкового віднесення вірусної програми до іншої модифікації цього ж класу, результат такого експеримент вважався як помилковим. Наприклад, якщо вірус Delf.a було віднесено до класу вірусу Delf з модифікацією b.

Слід відзначити, що запропонований метод виявлення на основі формування сигнатури поведінки програми може бути використаний для виявлення інших видів вірусних програм, зокрема нових версій існуючих вірусів. Для здійснення процесу виявлення нових сімейств, екземпляри вірусів повинні бути присутні у тестовій вибірці для навчання системи.

Таблиця 3

Ефективність виявлення вірусних програм

Назва вірусної програми	Варіант, що використовувався для навчання	Варіанти, що використовувалися для тестування	Кількість зразків для тестування	Accuracy, %
Delf	g	a	12	94
		d	24	95
		f	17	87
		h	10	89
		r	13	93
		Загалом	**76**	**92**
Bifrose	a	ae	21	88
		aq	18	93
		bg	18	95
		bh	14	93
		Загалом	**71**	**92**
MyDoom	c, h	a	12	93
		b	18	94
		g	22	93
		f	14	91
		Загалом	**66**	**93**

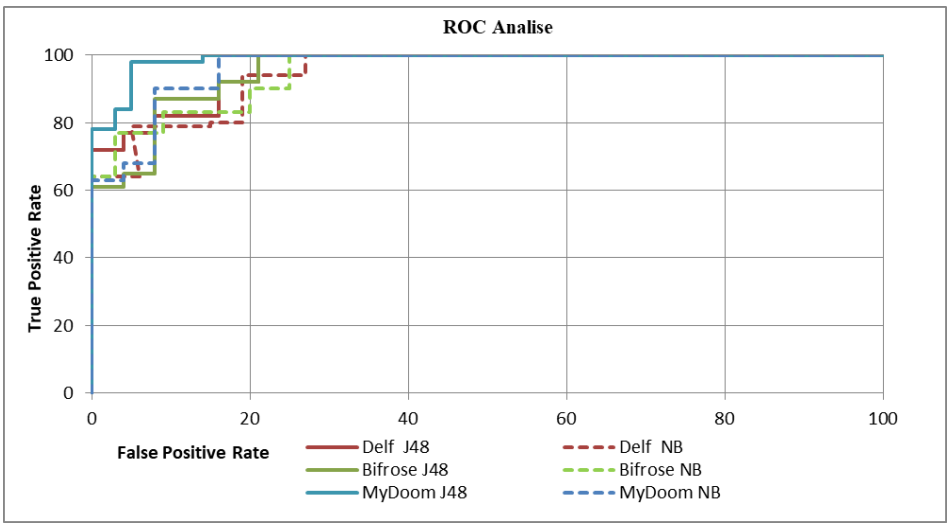

Рис. 5. ROC-криві для вірусних програм Delf, Bifrose та MyDoom (усереднено для всіх варіантів вірусних програм)

Висновок

В роботі представлено метод формування сигнатури вірусної програми на основі трасування API викликів, що дозволяє здійснити виявлення вірусної програми, яка представлена розробленою сигнатурою.

Для здійснення процесу виявлення вірусної програми представлено систему, що складається з трьох основних складових: сигнатура вірусної програми, база сигнатур і метод виявлення, що передбачає зіставлення сигнатур вірусу з базою сигнатур.

Сигнатура поведінки програми на основі трасування API викликів може бути представлена у вигляді сукупності двох складових: частота виклику та характер взаємодії критичних API викликів. Аналіз першої складової дозволяє визначити розподіл критичних API викликів за групами шкідливої активності та відображає кількісну складову сигнатури. Друга складова сигнатури передбачає відображення у векторний простір характеру взаємодії критичних API функцій вірусної програми та описує взаємозв'язок між критичними API функціями. Аналіз другої складової сигнатури надає можливість розмежувати вірусні програми від корисних додатків не тільки за наявністю критичних API викликів, але й за їх взаємодією між собою.

За результатами експериментальних досліджень загальна ефективність виявлення вірусних програм, що представлені розробленою сигнатурою, склала близько 92% з рівнем хибних спрацювань 6%.

Реалізація цього розробленого методу для проведення експериментальних досліджень здійснена в розподіленій багаторівневій системі виявлення зловмисного програмного забезпечення в локальних обчислювальних мережах.

Подальшим напрямом досліджень є розробка та наповнення бази сигнатур різних типів вірусних програм, а також доповнення розробленої системи поведінковими каркасами вірусів, наприклад з використання формальних методів, що дозволить здійснювати виявлення вірусних програм нульового дня.

Список використаної літератури

1. McAfee Labs Threat Report. December 2017 [Electronic resource] / Mode of access:, https://www.mcafee.com/us/resources/ reports/rp-quarterly-threats-dec-2017.pdf. – Last access: 2018. – Title from the screen.

2. Savenko, O. Approach for the Unknown Metamorphic Virus [Text] / O. Savenko, S. Lysenko, A. Nicheporuk, B. Savenko // Proceedings of the 9-th IEEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. – 2017. – P. 453–458.

3. Ki, Y. A Novel Approach to Detect Malware Based on API Call Sequence Analysis [Text] / Y. Ki, E. Kim, H. K. Kim // International Journal of Distributed Sensor Networks – Special issue on Advanced Big Data Management and Analytics for Ubiquitous Sensors. – 2015. – Vol. 2015. – No. 4. – P.120–129.

4. Lim, H. Detecting Malicious Behaviors of Software through Analysis of API Sequence k-grams [Text] / H. Lim // Computer Science and Information Technology. – 2016. – Vol. 4. – No. 3. – P. 85–91.

5. Sami, A. Malware Detection based on Mining API Calls [Text] / A. Sami, B. Yadegari, H. Rahimi, N. Peiravian // Proceedings of the 2010

ACM Symposium on Applied Computing. – 2010. – P. 1020–1025.

6. Alazab, M. Malware Detection Based on Structural and Behavioral Features of API Calls [Text] / M. Alazab, R. Layton, S. Venkataraman, P. Watters // Proceedings of the 1st International Cyber Resilience Conference. – 2010. – P. 1–10.

7. Sathyanarayan, V. S. Signature Generation and Detection of Malware Families [Text] / V. S. Sathyanarayan, P. Kohli, B. Bruhadeshwar // Proceedings of the 13th Australasian conference on Information Security and Privacy. – 2008. – P. 336–349.

8. Moldavskaya, A. V. Method of Learning Malware Behavior Scripts by Sequential Pattern Mining [Text] / A. V. Moldavskaya, V. M. Ruvinskaya, E. L. Berkovich // Symposium on Conformal and Probabilistic Prediction with Applications. – 2016. – P.196–207.

9. Christodorescu, M. Mining Specification of Malicious Behavior [Text] / M. Christodorescu, S. Jha, C. Krugel // Proceeding of the 6th joint meeting of the European Software Engineering Conference. – 2007. – P. 5–14.

10. Bergeron, J. Static Detection of Malicious Code in Executable Programs [Text] / J. Bergeron, M. Debbabi, J. Desharnais, M.M. Erhioui, Y. Lavoie, N. Tawbi // Symposium on Requirements Engineering for Information Security. – 2001. – P. 1–8.

11. MSDN Library [Electronic resource] / Mode of access: http://msdn.microsoft.com/ en-us/library/. – Last access: 2018. – Title from the screen.

12. VX Heavens Computer virus collection [Electronic resource] / Mode of access: http://vx.netlux.org. – Last access: 2018. – Title from the screen.

13. University Of Waikato. WEKA: Data Mining with Open Source Machine Learning Software [Electronic resource] / Mode of access: http://www.cs.waikato.ac.nz/ml/weka. – Last access: 2018. – Title from the screen.

14. API monitor [Electronic resource] / Mode of access: http://www.rohitab.com/apimonitor. – Last access: 2018. – Title from the screen.

15. Markowsky, G. Distributed System for Detecting the Malware in LAN [Text] / G. Markowsky, O. Savenko, A. Sachenko // Proceedings of the 2018 IEEEE 13th International Scientific and Technical Conference on Computer Science and Information Technologies. – 2018. – P. 306–309.

16. David, E. DeepSign: Deep Learning for Automatic Malware Signature Generation and Classification [Text] / E. David, N. S. Netanyahu // International Joint Conference on Neural Networks, Killarney, Ireland. – 2015. – P.1–8.

References

1. McAfee Labs Threat Report. December 2017, available at: https://www.mcafee.com/us/ resources/ reports/rp-quarterly-threats-dec-2017.pdf

2. Savenko, O., Lysenko, S., Nicheporuk, A., Savenko, B. (2017), "Approach for the Unknown Metamorphic Virus", *In. proc of the 9-th IEEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pp. 453–458.

3. Ki, Y., Kim, E., H. K. Kim (2015) "A Novel Approach to Detect Malware Based on API Call Sequence Analysis", *International Journal of Distributed Sensor Networks - Special issue on Advanced Big Data Management and Analytics for Ubiquitous Sensors*, Vol. 2015, No. 4, pp. 120–129.

4. Hyun-il Lim (2016), "Detecting Malicious Behaviors of Software through Analysis of API Sequence k-grams", *Computer Science and Information Technology*, Vol. 4(3), pp. 85–91.

5. Sami, A., Yadegari, B., Rahimi, H., Peiravian, N. (2010), "Malware Detection based on Mining API Calls", *In proc. of the 2010 ACM Symposium on Applied Computing (SAC)*, pp. 1020–1025.

6. Alazab, M., Layton, R., Venkataraman, S., Watters, P. (2010), "Malware Detection Based on Structural and Behavioral Features of API Calls", *In Proc. of the 1st International Cyber Resilience Conference*, pp. 1–10.

7. Sathyanarayan, V. Sai, Kohli, P., Bruhadeshwar, B. (2008), "Signature Generation and Detection of Malware Families", *In Proc. of the 13th Australasian conference on Information Security and Privacy*, pp. 336–349.

8. Moldavskaya, A. V., Ruvinskaya, V. M., Berkovich, E. L. (2016), "Method of Learning Malware Behavior Scripts by Sequential Pattern Mining", *Symposium on Conformal and Probabilistic Prediction with Applications*, pp. 196–207.

9. Christodorescu, M., Jha, S., Krugel, C. (2007), "Mining Specification of Malicious Behavior", *Proceeding of the 6th joint meeting of the European Software Engineering Conference*, pp. 5–14.

10. Bergeron, J., Debbabi, M., Desharnais, J., Erhioui, M.M., Lavoie, Y., Tawbi, N. (2001), "Static Detection of Malicious Code in Executable Programs", *In: Symposium on Requirements Engineering for Information Security*, pp. 1–8.

11. MSDN Library, available at: http://msdn.microsoft.com/ en-us/library/

12. VX Heavens Computer virus collection, available at: Availabe:http://vx.netlux.org

13. University Of Waikato. WEKA: Data Mining with Open Source Machine Learning Software, available at: http://www.cs.waikato.ac.nz/ml/weka

14. API monitor, available at: http://www.rohitab.com/ apimonitor

15. Markowsky, G., Savenko, O., Sachenko, A. (2018), "Distributed System for Detecting the Malware in LAN", *In Proc. of the 2018 IEEEE 13th International Scientific and Technical Conference on Computer Science and Information Technologies (CSIT)*, pp. 306–309.

16. David, E., Netanyahu, N. S. (2015) "DeepSign: Deep Learning for Automatic Malware Signature Generation and Classification", *International Joint Conference on Neural Networks*, Killarney, Ireland, pp.1–8.

FORMING OF THE PROGRAM'S BEHAVIOR SIGNATURE BASED ON THE API CALL TRACING

O. S. Savenko, A. O. Nicheporuk, A. A. Nicheporuk, Y. O. Nicheporuk
Khmelnytskyi National University

Abstract. *The paper proposes a method of forming a signature of a virus program based on API call tracing. The developed signature is compact and makes it possible to assign the virus program not only to one of the classes of viruses, but also allows determining its modification. An approach is proposed that allows detection of a virus program represented by a developed signature. The developed method for carrying out experimental researches is realized in distributed multilevel detection system of malicious software in local area computer networks.*

To implement the virus detection process, a system has been developed that consists of three main components: the signature of the virus program, the signature database and the method of detection, which involves comparing the signatures of the virus with the database of signatures.

Signature of the program's behavior based on API call tracing can be represented as a set of two components: the call frequency and the nature of the interaction of critical API calls. Analysis of the first component allows determining the distribution of critical API calls by groups of harmful activity and displays the quantitative component of the signature. The second component of the signature implies the mapping into the vector space the nature of the interaction of critical API functions that characterize virus program and describes the relationship between critical API functions. Analysis of the second component of the signature provides an opportunity to distinguish virus programs from useful applications not only in the presence of critical API calls, but also in their interaction with each other.

A number of experimental studies have been carried out to determine the accuracy of the proposed method for detecting virus programs using the developed signature. Viral family programs Delf, Bifrose and MyDoom were used as test data. The best detection rate was received for MyDoom virus programs (accuracy was 93%) with the use of the J48 binary classification algorithm.

Keywords: *virus program, API call tracing, chi-squared test, behavior*

ФОРМИРОВАНИЕ ПОВЕДЕНЧЕСКОЙ СИГНАТУРЫ ПРОГРАММЫ НА ОСНОВЕ ОБНАРУЖЕНИЯ API ВЫЗОВОВ

О. С. Савенко, А. О. Ничепорук, А. А. Ничепорук, Ю. О. Ничепорук
Хмельницкий национальный университет

Аннотация. *Предложено метод для формирования сигнатуры вирусной программы на основе трассировки API вызовов. Разработанная сигнатура позволяет осуществить отнесения вирусной программы не только к одному из классов вирусов, но и позволяет определять его модификацию. Разработанный метод для проведения экспериментальных исследований реализовано в распределенной многоуровневой системе обнаружения вредоносных программ в локальных сетях.*

Ключевые слова: *вирусная программа, трассировки API вызовов, хи-квадрат тест, поведение*

Отримано 13.10.2018

Савенко Олег Станіславович, кандидат технічних наук, професор, декан факультету програмування та комп'ютерних і телекомунікаційних систем Хмельницького національного університету. Вул. Інститутська, 11, Хмельницький, Україна, E-mail: savenko_oleg_st@ukr.net, тел. +38 067 9075315

Oleg Savenko, PhD, Professor, Dean of the Faculty of Programming and Computer and Telecommunication Systems, Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, Ukraine, E-mail: savenko_oleg_st@ukr.net, tel. +38 067 9075315

ORCID ID: 0000-0002-4104-745X

Нічепорук Андрій Олександрович, кандидат технічних наук, старший викладач кафедри комп'ютерної інженерії та системного програмування Хмельницького національного університету. Вул. Інститутська, 11, Хмельницький, Україна, E-mail: andrey.nicheporuk@gmail.com, тел. +38 096 4687613

Andrii Nicheporuk, PhD, Senior Lecture of Computer Engineering and System Programming Department, Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, Ukraine, E-mail: andrey.nicheporuk@gmail.com, tel. +38 096 4687613

ORCID ID: 0000-0002-7230-9475

Нічепорук Анастасія Андріївна, магістр кафедри комп'ютерної інженерії та системного програмування Хмельницького національного університету. Вул. Інститутська, 11, Хмельницький, Україна, E-mail: eldess06@gmail.com, тел. +38 098 4812570

Anastasia Nicheporuk, Master's Degree of Computer Engineering and System Programming Department, Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, Ukraine, E-mail: eldess06@gmail.com, tel. +38 098 4812570

ORCID ID: 0000-0002-8830-4256

Нічепорук Юрій Олександрович, аспірант кафедри комп'ютерної інженерії та системного програмування Хмельницького національного університету. Вул. Інститутська, 11, Хмельницький, Україна, E-mail: yuranichipor2015@gmail.com, тел. +38 096 3831136

Yuriy Nicheporuk, PhD Student of Computer Engineering and System Programming Department, Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, Ukraine, E-mail: yuranichipor2015@gmail.com, tel. +38 096 3831136

ORCID ID: 0000-0002-7648-3007