УДК 004.67

Савенко О. С., к.т.н.
Кльоц Ю. П. ,к.т.н.
Нічепорук А. О.,
Мостовий С. В.

ДІАГНОСТУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ НА НАЯВНІСТЬ МЕТАМОРФНИХ ВІРУСІВ НА ОСНОВІ МОДИФІКОВАНИХ ЕМУЛЯТОРІВ

Анотація. *Запропоновано метод діагностування комп'ютерних систем, який полягає у формуванні оцінки схожості копій метаморфного вірусу, що досягається шляхом створення на кожному хості свого модифікованого емулятора. Отриманні копії метаморфних вірусів порівнюються з використанням метрики Дамерау-Левенштейна. Для формування результату використовується система нечіткого логічного висновку.*

Ключові слова: *модифікований емулятор, метаморфний вірус, діагностування, метрика, нечіткий логічний висновок, локальна мережа, хост, опкод.*

Савенко О. С., к.т.н.
Кльоц Ю. П.,к.т.н.
Ничепорук А. А.,
Мостовой С. В.

ДИАГНОСТИКА КОМПЬЮТЕРНЫХ СИСТЕМ НА НАЛИЧИЕ МЕТАМОРФНЫХ ВИРУСОВ НА ОСНОВЕ МОДИФИЦИРОВАННЫХ ЭМУЛЯТОРОВ

Аннотация. *Предложено метод диагностики компьютерных систем, который заключается в формировании оценки сходства копий метаморфных вирусов, что достигается путем создания на каждом хосте своего модифицированного эмулятора. Полученные копии метаморфных вирусов сравниваются с использованием метрики Дамерау-Левенштейна. Для формирования результата используется система нечеткого логического вывода.*

Ключевые слова: *модифицированный эмулятор, метаморфный вирус, диагностирование, метрика, нечеткий логический вывод, локальная сеть, хост, опкод.*

Savenko O. S., PhD
Klots Y. P., PhD
Nicheporuk A.O.,
Mostovyy S.V.

COMPUTER SYSTEMS DIAGNOSTIC FOR THE METAMORPHIC VIRUSES BASED ON THE MODIFIED EMULATOR

Abstract: *The method of diagnosing computer systems is to create assessment of similarity metamorphic copies of the virus, which is achieved by creating for each host its modified emulator. Received metamorphic virus copy compared using metrics Damerau-Lowenstein. To generate results using fuzzy inference system.*

Keywords: *modified emulator, metamorfnyy virus, diagnostics, metrics, fuzzy logic, network, host, opcode.*

Вступ. Використання та поширення комп'ютерної техніки у всіх сферах сучасного життя супроводжується неспинним розвитком та поширенням комп'ютерних вірусів. Серед всієї множини вірусних програм одне з головних місць посідають метаморфні віруси. Наприклад, метаморфний вірус Sality у 2011 році здійснив інфікування близько 3 мільйонів комп'ютерів [1], а наприкінці 2014 року увійшов в п'ятірку найбільш розповсюджених вірусів (1,43% від загальної кількості всіх виявлених загроз) [2].

Попередні дослідження. Класичні методи виявлення вірусів, що засновані на сигнатурному аналізі, не здатні виявляти змінені копії метаморфного вірусу [3-5]. Для виявлення такого типу вірусів більшість антивірусних сканерів використовують евристичні методи виявлення, в якості характеристичних ознак яких виступають послідовності викликів API функцій, граф потоку управління програми, структурні особливості виконуваних файлів формату PE (Portable Executable)

.EXE, опкоди інструкцій (асемблерні команди) та їх комбінації.

Підхід в роботі [3] передбачає отримання машинних інструкцій шляхом дизасемблювання підозрілого файлу та розбиття його на функціональні підпрограми з подальшим порівнянням їх за допомогою метрики city blocks. Описаний підхід є не ефективним для вірусів, що використовують техніку перемішування блоків (code transposition), оскільки частота появи інструкцій не зміниться.

В [4] запропонований метод, що заснований на нормалізації коду для усунення основних технік обфускацій, що використовують метаморфні віруси. Висновок щодо шкідливості файлу здійснюється на основі визначення ізоморфізму підграфів у міжпроцедурному графі потоку управління. Проте, як відомо, задача визначення ізоморфізму графів відноситься до класу NP-повних задач, тобто зі збільшенням об'єму вхідних даних, різко зростає обчислювальна складність запропонованого методу.

У роботі [6] в якості сигнатури шкідливого коду пропонується побудова графу на основі взаємозв'язків системних викликів програми. Для спрощення обчислень всі системні виклики розділені на 32 класи. Для формування результату обчислюється схожість отриманих графів. Проте, для отримання всієї множини API викликів необхідне багаторазове виконання підозрілої програми.

Основним методом, що дозволяє здійснювати виявлення метаморфних вірусів є емуляції виконання [7], яка дозволяє проаналізувати підозрілий файл та виявити чи створюється підозрілий файл з обфускацією програмного коду. Проте, емуляція виконання є складним і трудоємнісним процесом, що окрім того, спричиняє навантаження на ресурси системи. Іншою проблемою яка ускладнює виявлення метаморфних вірусів є апріорна невизначеність характеристик, що можуть явно вказувати присутність метаморфних властивостей в програмному коді.

Метою статті є розробка нового методу виявлення метаморфних вірусів, що дозволить здійснювати віднесення підозрілого об'єкту до одного з класів метаморфних вірусів, з використанням модифікованих емулятори на кожному хості локальної мережі.

Матеріали дослідження. Розроблений метод використовує емуляцію виконання на кожному хості в мережі, використовуючи модифіковані емулятори.

Хости представляють собою мережні станції для обробки інформації, що поєднанні у локальну мережу. Основними функціями хостів є здійснення одноразової емуляції виконання невідомої програми та відправлення результатів на серверну частину.

Серверна частина слугує для опрацювання результатів виконання процесу емуляції, отриманих з хостів та здійснення нечіткого висновку про інфікування метаморфним вірусом. На рис. 1 зображено узагальнену схему виявлення метаморфних вірусів з використанням модифікових емуляторів.

Розглянемо детальніше розроблений метод.

Для виявлення підозрілих дій на кожному хості корпоративної мережі використовується аналізатор підозрілості програми.

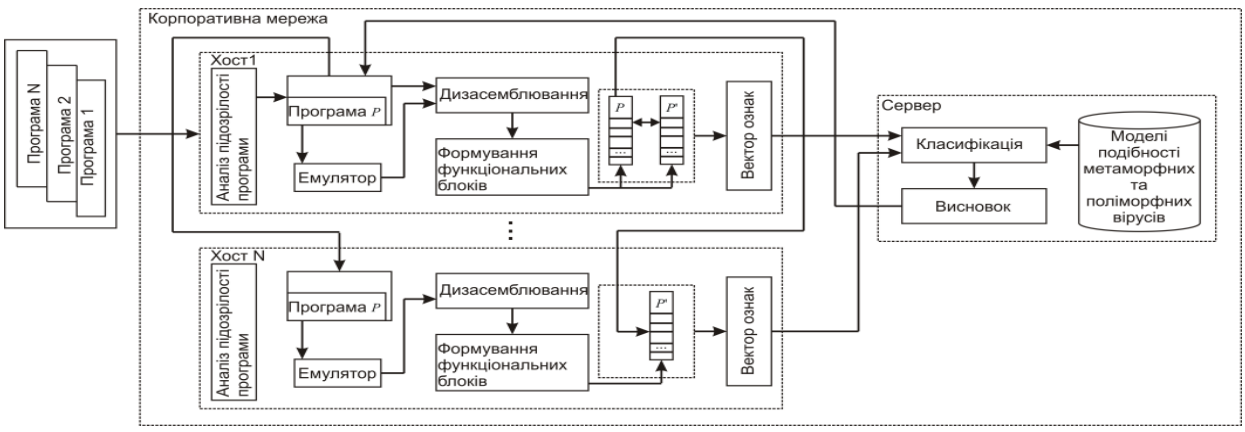

Рис 1. Узагальнена схема виявлення метаморфних вірусів з використанням модифікових емуляторів

Подамо вектор ознак, що визначає належність невідомої програми до потенційно небезпечного зразка коду:

$$\overline{U} = (M, Q, J, Y, L, N, H)$$

M – спроба програми отримати права адміністратора системи, Q – спроба відкриття або закриття системного порту, J – спроба видалення файлу, Y – створення файлу або процесу, L – перехоплення даних, що вводяться з клавіатури, N – розсилка повідомлень в мережу, H – створення або запис в системний реєстр.

Кожна ознака приймає значення 0 або 1, де 1 свідчить про активізацію відповідної ознаки, 0 – навпаки. Програму вважатимемо за підозрілу, якщо:

$$P = \text{suspicious}, \text{if } \forall u \in \overline{U}, (u_i = 1 \wedge u_j = 1)$$

Програма, для якої P=suspicious надходить в систему виявлення метаморфного коду.

Для отримання зміненого зразку коду F_S, здійснюється емуляція виконання програми P. Використання однотипного емулятора на всіх хостах мережі не дозволить з високим ступенем достовірності здійснити виявлення метаморфних вірусів, оскільки використання однакових емуляторів дозволить отримати лише однакові зразки коду. Тому, на кожному хості створюються модифіковані емулятори.

Структура емулятора наступна: віртуальний процесор, визначає набори інструкцій, що доступні для роботи (MMX, SSE, SSE2 та ін.) та включає в себе набори віртуальних регістрів; оперативна пам'ять та віртуальний стек; віртуальний мережний контролер; тип ОС (підтримка API-функцій, системного реєстру та портів) та модуль евристики.

Для протидії антиемуляційним технологіям, що використовуються у метаморфних вірусах, в емуляторі використовується модуль евристики. Для кожної операції, що виконується віртуальним CPU встановлюється фіксований час обробки та здійснюється перевірка повторів виконання певної операції (наприклад, якщо в тілі вірусу є цикл, що здійснює операцію інкременту змінної велику кількість разів).

З метою отримання початкового зразку коду F_P, здійснюється дизасемблювання програми P. Отриманий лістинг дизасембльованих інструкцій розбивається на функціональні блоки (ФБ), по інструкціях переходів.

Для пошуку схожості двох ФБ використовується дистанція Дамерау – Левенштейна. З використанням алгоритму поліноміальної складності Вагнера-Фішера [8].

Розглянемо програму P, яка складається з множини асемблерних команд p_i, тобто $P = \{p_1, p_2, ..., p_k\}$. Розіб'ємо програму P на функціональні блоки довільної довжини, що починаються із інструкцій умовного переходу jmp, jz та ін. і закінчуються ними, тобто $P = \{B_1, B_2, ..., B_l\}$. Тоді можна записати: $P = \{B_1 = \{p_1, p_2, ..., p_{i-1}\}, ..., B_l = \{p_i, p_{i+1}, ..., p_k\}\}$ Програму до емуляції позначимо F_P, а програму після емуляції виконання F_S.

Нехай функціональний блок B, що складається з множини опокдів, довжиною $|B| = m$ записується як $p_1, p_2, ..., p_m$, де p_i представляє i-й опкод p. Підмножина опкодів $x_i, x_{i+1}, ..., x_j$, функціонального блоку B буде позначатись B(i, j).

Вага перетворення опкода a в опкод b позначимо через $w(a, b)$. Таким чином, $w(a, b)$ – вага заміни одного опкоду на другий опкод, коли $a \neq b$, $w(b, a)$ – вага операції транспозиції, $w(a, \varepsilon)$ – вага видалення, а $w(\varepsilon, b)$ – вага вставки b.

Нехай B_g та B_h – два ФБ, що складаються з послідовності опкодів (довжиною n та m відповідно) над скінченим алфавітом асемблерних інструкцій $A = (a_1, a_2, ..., a_k)$, причому B_g ФБ програми F_P, позначимо $B_g^{F_P}$, а B_h – ФБ тієї ж програми після емуляції виконання F_S, позначимо $B_h^{F_S}$. Тоді відстань Дамерау – Левенштейна $dL(B_g^{F_P}, B_h^{F_S})$ обчислимо наступним чином:

$$OPT = \begin{cases} 0, & i = 0,\ j = 0 \\ i, & j = 0,\ i > 0 \\ j & i = 0,\ j > 0 \\ \min \begin{cases} OPT(i,\ j-1) + w(a,\varepsilon) \\ OPT(i,-1j) + w(\varepsilon,b) \\ OPT(i,-1,\ j-1) + w(a,b) \\ OPT(i-2,\ j-2) + w(b,a) \end{cases} & j > 0,\ i > 0 \end{cases}$$

Після отримання відстані Дамерау-Левенштейна для двох блоків B_g та B_h, формується зважене усереднене значення відповідного параметру вектора ознак для всіх блоків коду.

Для решти ознак (кількість операцій співпадіння, вставки, видалення, заміни, транспозиції) унормування відбувається аналогічним чином.

Подамо вектор ознак схожості копій метаморфних вірусів на основі метрики Дамерау-Левенштейна:

$$\bar{S} = \langle dL, T, D, I, R, M \rangle$$

де dL – відстань Дамерау – Левенштейна для функціонального блоку між програмами F_p та F_s ;T – кількість операцій обміну опкодів для перетворення блоку програми F_p у F_s ($F_p = F_s$);D – кількість операцій видалення опкоду; I – кількість операцій вставки опкоду; R – кількість необхідних операцій заміни відповідних опкодів; M – кількість співпадінь між опкодами в функціональному блоці програми F_p та F_s .

Отриманні вектори схожості копій метаморфних вірусів надходять на серверну частину для формування нечіткого логічного висновку.

Результати експериментів Для проведення досліджень з [9] було використано такі метаморфні генератори: Next Generation Virus Creation Kits, Second Generation Virus Generator та Virus Creation Lab for Win32.

Результатом роботи системи нечіткого логічного висновку є ступінь приналежності об'єкту до одного з чотирьох класів – трьох шкідливих та одного класу довірених додатків. Якщо ступінь приналежності невідомого об'єкту належить діапазону від 0 до 0,25, то невідомий об'єкт класифікується як довірений додаток; якщо ступінь приналежності лежить на проміжку від 0,26 до 1, то невідомий об'єкт відноситься до одного з класів метаморфних вірусів. На рис 2 наведено результати нечіткого логічного висновку для підозрілого файлу. В результаті 15% копій не змінились, 5% віднесено до першого класу метаморфних вірусів, 11,25 – до третього та 68, 75% до другого.

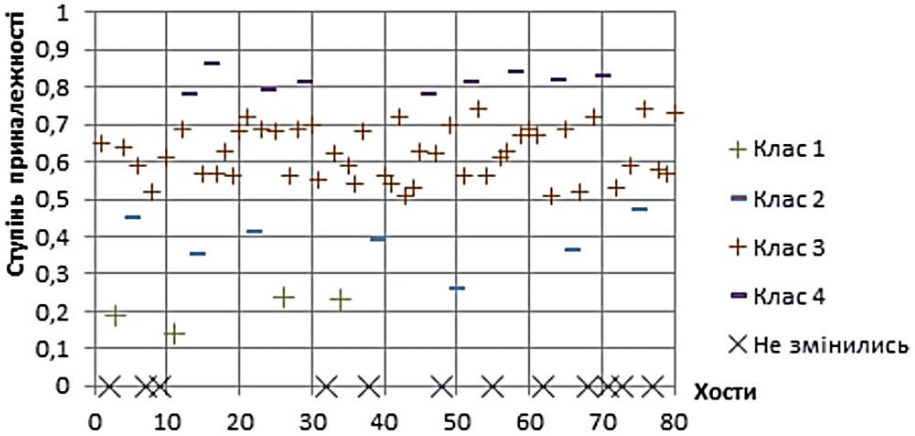

Рис 2. Результати нечіткого логічного висновку

Висновки. Аналіз предметної області показав необхідність у вдосконалені існуючих методів виявлення метаморфних вірусів. Авторами запропоновано метод виявлення метаморфних вірусі з використанням модифікованих емуляторів на хостах локальної мережі. Класифікація вірусів здійснюється на основі системи нечіткого логічного висновку. Технологія дозволяє віднести досліджуване програмне забезпечення до одного з класів метаморфних вірусів, в яких застосовано техніку обфускації програмного коду. Результати

експериментальних досліджень продемонстрували ефективність виявлення метаморфних вірусів на рівні 85%.

Список використаної літератури

1 Falliere N. Sality: Story of a Peer-to-Peer Viral Network [Text] / N. Falliere // Symantec Labs. – p. 1-21.

2 Вирусная статистика. Обзор киберугроз ноября 2014 года [Електронний ресурс]. – режим доступу: https://www.esetnod32.ru/company/viruslab/statistics/?id= 896397.

3 Rad, B. B. Metamorphic Virus Variants Classification Using Opcode Frequency Histogram [text] / B.B. Rad, M. Masrom // 14th WSEAS international conference of computers. – Wisconsin. – p. 147-155.

4 Bruschi, M. M. D. Detecting self-mutating malware using control-flow graph matching [text] / M.M.D. Bruschi, M. Martignoni, L. Monga // Journal Detection of Intrusions and Malware & Vulnerability Assessment. – Berlin. – 2006. – p. 129-143.

5 Attaluri S. Profile Hidden Markov Models and Metamorphic Virus Detection [text] / S. Attaluri, S. McGhee, M. Stamp // Journal in Computer Virology. – 2009. – V.5. – p. 151-169.

6 Lee J. Detection metamorphic malware using code graphs [text] / J. Lee, H. Jeong, H.Lee // Proceedings of the 2010 ACM symposium on applied computing. – New-York. – 2010. – p. 1970-1977.

7 Szor P. Hunting for Metamorphic [text] / P.Szor // VirusBulletin. – 2001. – p. 123-144.

8 Wagner R., Fisher, M. The string to string correction problem [text] / R. Wagner, M. Fisher // Journal of the ACM. – 1974. –V.21. –p. 168-178.

9 Коллекция вирусов VX Heavens [Електронний ресурс]. – режим доступу: https://vxheaven.org/vl.php?lang=ru.

Отримано 00.00.2016

References

1 Falliere, N.: Sality: Story of a Peer-to-Peer Viral Network. *Technical report, Symantec Labs*, pp. 1-21 (in English).

2 Faylovyy virus Win32.Sector zarazil bolee milliona komp'yuterov – Retrieved from http://news.drweb.ru/show/?i=5778 . – 21.05.2014 (in Russian).

3 Rad, B. B., Masrom, M.: Metamorphic Virus Variants Classification Using Opcode Frequency Histogram, (2010), *14th WSEAS international conference of computers*, pp. 147-155 (in English).

4 Bruschi, M. M. D., Martignoni, L.,Monga, M.: Detecting self-mutating malware using control-flow graph matching, (2006), *Proc. of the Conference on Detection of Intrusions*, Berlin, Germany, pp. 129-143 (in English).

5 Attaluri, S., McGhee, S., Stamp, M.: Profile Hidden Markov Models and Metamorphic Virus Detection, (2009), *Journal in Computer Virology*, Vol.5, pp. 151-169 (in English).

6 Lee, J., Jeong, H., Lee, H. Detection metamorphic malware using code graphs,(2010), *Proceedings of the 2010 ACM symposium on applied computing*, pp. 1970-1977 (in English).

7 Szor, P., Ferrie, P. Hunting for Metamorphic, (2001), *VirusBulletin*, pp. 123-144 (in English).

8 Wagner, R., Fisher, M. The string to string correction problem, (1974), *Journal of the ACM*, 21, pp. 168-178 (in English).

9 VX Heavens Computer virus collection. Availabe: http:// vxheaven.org

Савенко
Олег Станіславович
к.т.н., доц. каф. системного програмування Хмельницького національного університету
т. +38(067)9075315
savenko_oleg_st@ukr.net

Кльоц
Юрій Павлович
к.т.н., доц. каф. системного програмування Хмельницького національного університету
т. +38(067)7740840
sprklyots@gmail.com

Нічепорук
Андрій Олександрович
асистент каф. системного програмування Хмельницького національного університету
т. +38(096)4687613
andrey.nicheporuk@gmail.com

Мостовий
Сергій Володимирович
ст. викл. каф. системного програмування Хмельницького національного університету
т. +38(096)4687613
sprmostovuy@gmail.com