УДК 004.92

О. С. САВЕНКО, С. М. ЛИСЕНКО, А.О. НІЧЕПОРУК

Хмельницький національний університет, Хмельницький

МОДЕЛЬ ПРОЦЕСУ ДІАГНОСТУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ НА НАЯВНІСТЬ ПОЛІМОРФНОГО ТА МЕТАМОРФНОГО ПРОГРАМНОГО КОДУ

Анотація. Дана робота присвячена проблемі виявлення поліморфних та метаморфних вірусів. Зокрема, в роботі проаналізовано методи їх виявлення. Було встановлено, що існуючі методи не здатні в повному обсязі проводити виявлення таких типів вірусних програм. Шляхом аналізу особливостей функціонування поліморфних та метаморфних вірусів, нами було запропоновано узагальнену модель поліморфних вірусів, що ґрунтується на основі їх поведінки при інфікуванні файлу. На основі моделі поліморфних вірусів було запропоновано модель процесу діагностування комп'ютерних систем на наявність поліморфного та метаморфного програмного коду.

Ключові слова: поліморфний вірус, відбиток, поведінка вірусу

Аннотация. Данная работа посвящена проблеме обнаружения полиморфных и метаморфных вирусов. В частности, в работе проанализированы методы их обнаружения. Было установлено, что существующие методы не способны в полном объеме проводить выявление таких типов вирусных программ. Путем анализа особенностей функционирования полиморфных и метаморфных вирусов, нами было предложено обобщенную модель полиморфных вирусов, основаной на их поведении при инфицировании файла. На основе модели полиморфных вирусов была предложена модель процесса диагностирования компьютерных систем на наличие полиморфного и метаморфного программного кода.

Ключевые слова. полиморфный вирус, отпечаток, поведение вируса

Abstract. This paper deals with the problem of detection of polymorphic and metamorphic viruses. In particular, the paper analyzes the methods of detection. It was found that the existing methods are not able to fully carry out these types of virus detection software. By the analysis of the functioning of polymorphic and metamorphic viruses, we proposed a generalized model of polymorphic viruses based on their behavior when infected file. Based on the model of polymorphic viruses were proposed model of computer systems diagnosing the presence of polymorphic and metamorphic code.

Key words: polymorphism, pattern, behavior of the virus.

Вступ

На сьогодні всі обчислення, ведення обліку, контролю не можливі без використання комп'ютерних систем. У більшості операційних систем програми можуть модифікувати інші програми та /або файли. Таку можливість, окрім звичайних програм, використовують віруси, зокрема поліморфні та метаморфні [1,2], що значно ускладнює процес розпізнання поліморфної чи метаморфної програми.

Актуальність

Поліморфні віруси відносять до класу шифрованих вірусів (хоча деякі поліморфні віруси можуть бути незашифрованими, наприклад вірус BadBoy [1]), проте окрім підпрограми розшифровки та зашифрованого тіла вірусу, поліморфні віруси містять ще один модуль – блок мутації. Блок мутації генерує довільним чином процедуру розшифрування основного тіла вірусу, що змінюється кожного разу, коли поліморфний вірус втілюється в інші файли. Таким чином при кожному новому інфікуванні файлу сигнатура поліморфного вірус буде відрізнятися від покоління до покоління, що значно ускладнює процес виявлення таких вірусів.

За даними MacAfee у 2013 році 58% всіх ботнет атак, метою яких була крадіжка фінансових даних, припадала на поліморфний ботнет Zeus [5]. Тому розроблення процесу виявлення та ідентифікації поліморфних та метаморфних вірусів є актуальним завданням.

Мета

Метою написання даної статті є аналіз властивостей поліморфних та метаморфних вірусів, їх життєвого циклу та створення узагальненої моделі процесу діагностування поліморфних та метаморфних програм.

Задачі

1. Проаналізувати механізми функціонування поліморфних та метаморфних вірусних програм.
2. Побудувати узагальнену модель функціонування поліморних вірусів
3. Розглянути модель процесу діагностування комп'ютерних систем на наявність поліморфних та метаморфних вірусних програм.

Розв'язання задач

Поліморфні віруси – це шкідливе програмне забезпечення, в якого окрім тіла вірусу, що виконує деструктивні дії, та програми розшифровки, що безпосередньо розшифровує основне тіло вірусної програми, присутній блок мутації, для зміни програми розшифровки при кожному новому інфікуванні файлів. Загалом будь-який поліморфний (метаморфний) вірус повинен пройти ряд стадій.

Знаходження власного коду: кожного разу, коли поліморфний двигун трансформує код, він повинен бути спроможним знайти власний код у нових варіантах;

Розшифрування: на наступному етапі двигун повинен розшифрувати інформацію, що потрібна для виконання перетворень. Для того щоб трансформувати себе, двигун повинен мати деяке уявлення про себе, щоб він знав, як зробити перетворення.

Аналіз: для коректної роботи поліморфних перетворень повинна бути доступна певна інформація. Якщо така інформація відсутня, сам поліморфний двигун повинен побудувати її. Зокрема, частиною інформації, яка часто потрібно для аналізу і трансформації є граф потоку управління (CFG) програми.

Перетворення: даний блок відповідає за перетворення коду в еквівалентний код. Це досягається зазвичай заміною на еквівалентні інструкції.

Приєднання: Останній крок, це приєднання нової генерації вірусу до нового файлу-хоста.[4]

Звісно, полморфний вірус для свого функціонування може пройти всі описані вище стадії або лише частину з них. Проте стадії знаходження власного коду, аналіз та перетворення повинні пройти всі поліморфні віруси.

Для виявлення таких типів вірусів використовуються методи емуляції програмного коду вірусу, що дозволяють відслідковувати можливі прояви його поведінки.

Існуючі підходи для виявлення поліморфних вірусів, що базуються на основі емуляції можна поділити на дві частини – статичні та динамічні.

Статичний аналіз це процес аналізу програмного коду без його фактичного виконання. Під час виконання цього процесу файл, що інфікований вірусом, зазвичай дизасемблюється першим. Тоді, застосовуються методи для аналізу функціонування програми на основі аналізу потоку управління та потоку даних, що отримані на етапі дизасемблювання. Зокрема у роботах [3,4] було розглянуто та запропоновано ряд статичних методів.

У [3] виявлення грунтується на основі побудови абстрактного керуючого графу інфікованого файлу за допомогою зовнішніх предикатів, що є результатами множини статичних аналізаторів.

Іншим підходом є побудовою матриці суміжності для керуючого графу з створенням 16 бітного відбитку (pattern) [4].

Основною перевагою статичних методів є те, що вони здатні покрити весь код програми і, як правило, швидші ніж динамічні. До недоліків даної групи методів можна віднести те, що код, що аналізується, не обов’язково може бути кодом, який насправді зараз виконується. Даний недолік особливо гостро проявляється при аналізі поліморфних та метаморфних вірусів. Окрім того, поліморфні віруси використовують методи обфускації для перешкоджання процесу дизасемблювання. Це впершу чергу пов’язано з особливостями деяких наборів інструкцій для архітектур x86 та x64 де важко виявити різницю між байтами даних та кодом у файлі.

Динамічні методи аналізують код під час виконання. В порівнянні із статичними методами динамічні проводять аналіз тих інструкцій, що виконуються в цей момент часу. За допомогою даного підходу можна виявляти код, що самомодифікується і проводити виявлення обфускованих методів. Але використання даного підходу на не захищеному комп’ютері може призвести до зараження інших файлів.

Враховуючи особливості функціонування поліморфних комп’ютерних вірусів побудовано модель процесу виявлення та розпізнання типу поліморфних вірусів.

Згідно із класифікацією [1] розрізняють 6 рівнів поліморфних вірусів. В роботі [2] подані моделі рівнів поліморфних вірусів. Приймемо узагальнену модель поліморфного вірусу кортежем:

$$M_{\rho} = \langle A, B, X, G, U, \xi, H, D, R \rangle . \quad (1)$$

де A - множина команд певної програми, яка може бути інфікована вірусом, $A = \{a_1, ..., a_n\}$; V – множина команд вірусу для вибору одного з присутніх у вірусі розшифровувачів, $V = \{v_1, ..., v_m\}$; X - множина розшифровувачів присутніх у вірусі, $X = \{x_1, ..., x_y\}$; G – множина команд вірусу xi розшифровувача, $G = \{g_1, ..., g_{\theta}\}$; U - множина шкідливих команд (тіло вірусу), $U = \{u_1, ..., u_w\}$; B - множина «команд-сміття», $B = \{b_1, ..., b_h\}$; ξ - функція вибору x_i розшифровувача, $\xi : V \to X$; H – функція утворення шкідливих команд засобами вибору x_i розшифровувача командами $g_{x_i} \in G$ і генерації порядку їх виконання, $H : B \times G_{x_i} \to U$; D - функція утворення поведінки поліморфного вірусу R

шляхом вкорінення шкідливих команд U в команди програми A, $D: A \times U \to R$; функція утворення поведінки поліморфного вірусу R без вкорінення у певну програму шляхом розшифрування одним з розшифровувачів x_i командами $g_{x_i} \in G$ і генерації порядку їх виконання матиме вигляд: $D: U \to R$.

Поведінки вірусу п'ятого рівня поліморфізму R_5^A та R_5 можна отримувати послідовності вигляду: $R_5^A = g_{\phi_{x_\varepsilon}} ... g_{\eta_{x_\varepsilon}} a_1 ... a_n u_\vartheta b_\rho ... u_\sigma b_\zeta$; $R_5^A = g_{\phi_{x_\varepsilon}} ... g_{\eta_{x_\varepsilon}} u_\vartheta b_\rho ... u_\sigma b_\zeta$, де значення ρ, ζ визначають, що можливі вірусні команди розшифровувача $g_{\phi_{x_\varepsilon}} ... g_{\eta_{x_\varepsilon}}$ можуть бути різними для різних розшифровувачів x_ε, ϑ - номер обраного розшифровувача, значення $\rho, \zeta, \vartheta, \sigma$ визначають, що можливі «команди-сміття» і команди вірусу $u_\vartheta b_\rho ... u_\sigma b_\zeta$ можуть бути різними для кожного нового запуску вірусу.

Отримана узагальнена модель поліморфного вірусу дозволяє побудувати модель процесу діагностування комп'ютерних систем на наявність поліморфного програмного коду. Весь процес діагностування КС на наявність поліморфного коду складається з двох етапів: процесу моніторингу (множина S) та процесу розпізнання поліморфного вірусу (множина Ψ).

$$M = \langle S.\Psi, A, P_{\text{inf}}, R, M, \rho \rangle . \tag{2}$$

де S – процес сканування забезпечує моніторинг активності файлової системи КС, $S = \{S_1, ..., S_5\}$. Зокрема під моніторинг файлової системи підлягають такі ресурси, як активність процесів, доступ процесів до API функцій, що можуть створювати та змінювати процеси, зокрема функція CreateProcess. Для ефективного виявлення необхідною умовою є активність процесу сканування зі запуском КС. При виявленні підозрілих дій об'єкт діагностування надходить на вхід процесу розпізнання. Процес сканування складається з етапів:

S_1 – процедура моніторингу доступу до системних функцій КС, зокрема зміна системного реєстру, виклик API функцій, що входять в список підозрілих, тобто API функції спрямованні на безпосередню зміну структури файлу, зміна завантажувальних секторів жорсткого диску. Так при аналізі API функцій виконуваного файлу типовим набором є наприклад поява GetVersion, GetCommandLineA. Тому поява API функцій типу CreateFileA, RegOpenKeyA, RegOpenKeyExA, LoadLibraryA, FindFirstFileA, FindFirstFileExA можна вважати потенційно небезпечними, оскільки вони дозволяють здійснювати потенційно небезпечні дії;

S_2 – процедура моніторингу інформаційних потоків даних через системні порти КС, для контролю завантаження веб сторінок з інтернету та обміну повідомлення і файлами засобами електронної почти;

S_3 – процедура сканування на предмет контроллю цілісності файлів (на базі CRC32). Процедура S_3 формує ідентифікатор файлу та сканує КС на предмет зміни ідентифікатора. Ідентифікатор файлу складається з інформації про атрибути, дати створення, розміру та контрольної суми CRC32 або MD5.

S_4 – виконання функції підозрілості по відношенню до об'єкту діагностування для визначення ступеня небезпеки інфікування поліморфним вірусом КС. Вхідними даними для функції S_4 є вихідні дані процедур S_1, S_2 та S_3 (табл.1). Дана функція забезпечує формування оцінки небезпечності файлу. Так, імовірність інфікування КС поліморфним вірусом, що використовує всі ознаки становить $P_{complex}$ =1, тоді імовірність інфікування КС поліморфним або метаморфним вірусом, що залежить від імовірності виникнення подій в системі становить:

$$P_{\text{inf}} = \sum_{i=1}^{n} \mu \frac{P_{a_i^j}}{P_{complex}} \tag{3}$$

Під коефіцієнтом небезпечності події в КС μ будемо розуміти характеристичну міру, яка визначає важливість тієї чи іншої події, що відбулася в КС. Оберемо коефіцієнт μ, такий що $\mu = \overline{1,2}$, тобто поява події в якої $\mu = 2$ означає більшу вагу.

Будемо вважати бар’єром імовірності, при якому $P_{\text{inf}} \geq 0{,}54$. Тобто досягнення цієї імовірності означатиме потенційну загрозу інфікування і перехід до на наступної стадій розпізнання поліморфного вірусу.

S_5 – виконання функції блокування або додавання до “білого списку” підозрілого об’єкту на основі ступеня підозрілості, що отримана з попередніх процедур. Білим списком вважається певний набір файлів бар’єр імовірності яких перевищує 0,54, тобто $P_{\text{inf}} \geq 0{,}54$. Проте дані файли не відносяться до інфікованих. Наприклад деякі програми для приховування власного коду від його розбору використовують поліморфізм.

Ψ – процес розпізнання, $\Psi = \{\Psi_1, \Psi_2, \Psi_3\}$. Визначає належність можливо інфікованої програми до поліморфного вірусу, і якщо це підтверджується, то визначається належність до одного з шести рівнів поліморфізму.

Процес розпізнання поліморфного коду складається з етапів:

Ψ_1 – виконання емуляції підозрілого об’єкта для його виконання в віртуальному середовищі, що імітує реальну роботу ОС;

Ψ_2 – функція отримання відбитку підозрілого файлу. Для отримання відбитку вірусу, розглянемо файл F розміром n. Він описує потенційно інфікований зразок або шкідливий зразок. F є послідовністю байтів, або іншими словами послідовністю, яка складається з k символів, $\sum = K_{255} = \{0,1,...,255\}$, отже F є елементом $\sum^k$.

Опишемо P_Π відбиток ПКВ розміром s, по відношенню до даного ПКВ Π як кінцевої послідовності з $\sum^s$. Таким чином:

$P_\pi = \{b_1, b_2, ..., b_s\}$, і-й байт F (відповідно P_Π), позначимо F(i) (відповідно $P_\Pi(i)$).

Можна стверджувати, що файл F інфікований ПКВ Π, якщо його s міститься у F і позначимо $\{i_1, i_2, ..., i_s\}$, такі що $F(i_j) = b_{\sigma(j)}$,

де $1 \leq j \leq s$, σ – взаємооднозначне відображення мутації байтів P_Π. Використання перестановок σ дозволяє врахувати можливі перестановки даних ПКВ у файлі F.

Згідно з цим у випадку з ПКВ можливі варіанти:

Зміна індексації байтів P_Π, тоді $\sigma = ID_{N_s}$;

Зміна порядку слідування байтів, тоді $\sigma \neq ID_{N_s}$.

Ψ_3 – виконання функції порівняння отриманого відбитку підозрілого файлу з поведінковими послідовностями, що дозволяють визначити рівень поліморфного вірусу (1).

Для представлення кожного етапу необхідні відповідні аргументи, що реалізуються за допомогою множини параметрів $\bigcup_5 A_i = \{a_1, ..., a_8\}$, де кожен параметр складається з множини вхідних значень, що характеризують кожний етап $A_i = (a_1^i, a_2^i, ..., a_{k_i}^i)$, де і – аргумент, $i = \overline{1,5}$, а k_i кількість вхідних значень для і-го аргументу.

Результатом процесу діагностування R, $R : S \times \Psi$ є визначення інформації про характер інфікування поліморфним вірусом КС.

На рис. 1 зображена модель процесу діагностування КС на наявність поліморфного та метаморфного програмного коду.

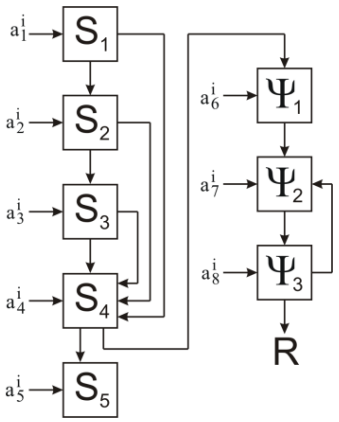

Рисунок 1 – формалізована схема процесу діагностування КС на наявність поліморфного та метаморфного програмного коду

Таблиця 1 – Параметри і їх поява в КС

Параметри системи	Відслідковування параметрів реєстру a_1^1	Виклик потенційно небезпечних API функцій a_1^2	Зміна завантажувального сектора HDD a_1^3	Дата першої перевірки системних портів* a_2^1	Моніторинг системних портів a_2^2	Дата першої перевірки атрибутів, дати, створення, розміру та контрольної суми файлу* a_3^1	Формування ідентифікатора файлу a_3^2
К-нт небезпечності події, μ	1	1	2	2	1	2	2
Імовірність появи події, P_{a_i}	0,09	0,09	0,18	0,18	0,09	0,18	0,18
Поява події	0	0	0	0	0	0	0
	0	0	0	0	0	0	1
	…	…	…	…	…	…	…
	1	1	1	1	1	1	1

Примітка: * При умові, що дата першої перевірки вже була здійснена при початковому скануванні КС.

На рис.2 зображено залежність інфікування КС від активації параметрів системи. Так, наприклад, поліморфний вірус Alman.a, що виконує крадіжку персональних даних відкриває порт UDP 137, створює ключ реєстру [HKCR\CLSID\{C111980D-B372-44b4-8095-1B6060E8C647}], викликає API функції CreateFileA, та RegOpenKeyA, додає в початок файлу 34 Кб, що спричиняє зімну контрольної суми та ідентифікатора файлу. Його $P_{\text{inf}} = 0{,}63$, що є більшим за 0,54. Натомість програма test.exe створює лише ключ реєстру та викликає API функцію CreateFileA.

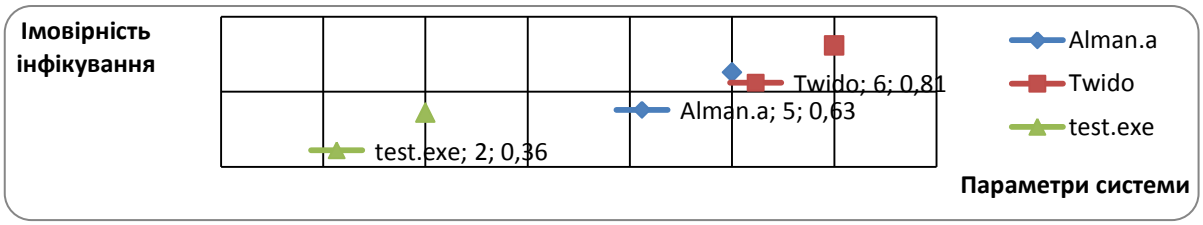

Рисунок 2 – Залежність імовірності інфікування КС від активації параметрів системи

Висновок

На основі дослідження було проаналізовано рівні поліморфних комп’ютерних вірусів, що дозволило отримати модель процесу виявлення поліморфних та метаморфних вірусів шляхом формування критичних параметрів системи та формування імовірності їх появи в КС. Отримана модель процесу діагностування дозволяє ідентифікувати рівні поліморфного вірусу, що інфікував систему.

Список посилань

1. Szor, P. Hunting for Metamorphic / P. Szor, P. Ferrie// VirusBulletin. -Sept, 2001. -P. 123-144.

2. Pomorova O. A technique for detection of bots which are using polymorphic code / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk and A. Nicheporuk. A// Kwiecien A., Gaj, P., Stera, P. (eds.) CN2013. – Springer, – 2014 . – p. 265-276/.

3. Christodorescu M. Semantics-Aware Malware Detection / M. Christodorescu, S. Jha, S.A. Seshia, D.Song, Bryant, R.E. // Proceedings of the 2005 IEEE Symposium on Security and Privacy, 2005, Washington, DC, USA, 2005. – p. 32-46.

4. Christodorescu M. Static Analysis of Executables to Detect Malicious Patterns / M. Christodorescu, S. Jha // 12th conference on USENIX Security Symposium, Aug., 2003, Berkeley, USA, 2003. – p. 169-186.

5. Zeus retains botnet crown, according to McAfee [Електронній ресурс] / Режим доступу http://www.v3.co.uk/v3-uk/news/2258398/zeus-still-king-of-the-botnets-say-researchers (дата звернення 21.09.2014).

Відомості про авторів

Савенко Олег Станіславович – к.т.н., доцент, декан ФПКТС, доцент кафедри системного програмування, Хмельницький національний університет.

Лисенко Сергій Миколайович – к.т.н., доцент, доцент кафедри системного програмування, Хмельницький національний університет.

Нічепорук Андрій Олександрович – аспірант, Хмельницький національний університет.