УДК 621.391 160164

С.М. ЛИСЕНКО, В.В ГУМЕНЮК
Хмельницький національний університет

МЕТОД ВИЯВЛЕННЯ ШКІДЛИВИХ ПРОГРАМНИХ ЗАСОБІВ НА ОСНОВІ АЛГОРИТМУ НАЙБЛИЖЧИХ СУСІДІВ

В роботі представлено метод виявлення шкідливого програмного забезпечення на основі алгоритму k-найближчих сусідів, який здійснює класифікацію програмного забезпечення на шкідливе і нормальне. Метод передбачає аналіз поведінки програмного забезпечення в комп'ютерній системі. В процесі роботи методу кожен досліджуваний процес представляється у вигляді вектора, у якому кожен елемент є значення статистичного показника, що відображає кількість системних викликів, які процес здійснює на протязі його виконання. Метод дозволяє забезпечити реагування на нові загрози, забезпечуючи захист комп'ютерних систем від як відомого так і невідомого ШПЗ. Робота системи виявлення ШПЗ здійснюється на основі виявлення аномалій через порівняння поведінки досліджуваного процесу із базою даних відомих поведінок програмного забезпечення в комп'ютерних системах.

Ключові слова: алгоритм k-найближчих сусідів, шкідливе програмне забезпечення, категоризація, системні виклики,.

S.M. LYSENKO, V.V. GUMENYUK
Khmelnytskyi National University, Khmelnytskyi, Ukraine

APPROACH FOR MALICIOUS SOFTWARE DETECTION USING *K*-NEAREST NEIGHBORS ALGORITHM

Abstract – this article describes a method for malware detection based on k-nearest neighbors algorithm (kNN) through binary classification of processes as malicious or benign.

The still-dominant method of malware detection is signature detection. This method relies on detecting unique patterns of data inside of scanned files. The rate of appearance of new malware specimens tends to outpace malware specialists' ability to create new signatures. This means that signature-based detection methods suffer from low detection rate of new types of malware. The field of machine learning provides more reliable means of protection against such threats.

The method described in this article employs k-Nearest Neighbor classification algorithm, successfully employed in the field of text classification, for the task of detection of malicious software. This method treats each process as a document to be classified. Feature vectors extracted from each process contain weighted frequencies of each system call performed during the process' runtime. Each unknown processes is compared with processes from a database of known benign processes, which is to be formed on the training phase. When the average similarity of k most similar processes reaches the certain threshold this process can be classified as benign, otherwise – as malicious. For the measuring of the similarities of feature vectors, cosine similarity was used. Two weighting methods: tf-idf and frequency for extraction of feature vectors were compared. It was shown that performance of this method depends on the choice of weighting method, the value of k and the threshold value. Different trade-offs can be achieved by varying these parameters. It was shown that tf-idf weighting method provides the optimal balance between detection rate and false positive rate. On the other hand, frequency weighting provides higher detection rate and the ease of forming the training database at the cost of much higher false positive rate. It makes sense to investigate different weighting methods and measures of similarity further. Experiments have shown applicability of the proposed method and the possibility of implementing it in software at the detection rate of 96-98% with the false positive rate of 3-6%.

This method allows for detection of new, previously unknown types of malware, providing protection of computer systems from both known and unknown malware.

Keywords: k-Nearest neighbor classifier, malware, system calls, attack, document categorization.

Вступ

Захист комп'ютерних систем є критично важливим і актуальним завданням, враховуючи складності виявлення загроз в реальному часі. Найсучасніші методи захисту не здатні в повній мірі забезпечити високий рівень ефективності та достовірності виявлення шкідливого програмного забезпечення (ШПЗ). Багато галузей індустрії покладаються на антивірусні засоби, які ефективні проти загроз із відомими сигнатурами, але які не здатні виявляти нові загрози, що з'являються щоденно та неефективні проти багаточисленних 0-day та поліморфного ШПЗ. [1]

Підходи на основі аналізу поведінки намагаються виявити ШПЗ на основі аналізу виконання ними послідовностей інструкцій чи системних викликів [2-4]. Ці підходи в основному покладаються на традиційні моделі машинного навчання, такі як приховані Марковскі моделі, метод опорних векторів, дерева рішень. Тим не менш, існуючі методи на основі машинного навчання характеризуються високим рівнем хибних спрацювань. Ситуація пояснюється різноманітності та складності сучасного програмного забезпечення, як шкідливого так і безпечного, неоптимального вибору ознак аналізу поведінки ПЗ, обмеженими (або застарілими) наборами даних.

Оскільки ШПЗ постійно еволюціонує, існуючі механізми захисту не справляються із зростаючою складністю загроз. Більш того, уся множина ШПЗ не є однорідною – складність ШПЗ залежить від цілі, типу служби, що підлягає атаці, джерела атаки, та місця знаходження цілі.

Вразливості комп'ютерних систем виникають внаслідок недолік політик безпеки, поганої конфігурації систем чи недосконалих програм. Системи виявлення ШПЗ грають вагому роль в знаходженні атак, що використовують ці вразливості чи недоліки.

Ідеальна система безпеки – це така, що має 100% успіх у виявлені ШПЗ та 0% хибних спрацювань (коефіцієнт неправильної класифікації нормальної поведінки), не вимагає ручного моніторингу та не використовує занадто багато системних ресурсів. Проте, відомі системи виявлення ШПЗ мають високий рівень хибних спрацювань та низький рівень виявлення загроз.

Існує два загальних підходи до виявлення ШПЗ: виявлення за сигнатурами та виявлення аномалій. Виявлення за допомогою є корисним для виявлення відомого ШПЗ, однак такий підхід неефективний проти нових видів шкідливого програмного забезпечення. На відміну від виявлення за сигнатурами, виявлення аномалій виявляє поведінки ПЗ, що суттєво відрізняються від відомих. Для забезпечення виявлення шкідливої поведінки в комп'ютерній системі (КС) першою задачею є побудова множини нормальних поведінок. З цією метою доцільним є використовуються технологій машинного навчання, які в подальшому здатні класифікувати нову поведінку на нормальну чи аномальну ([4]). Незважаючи на їх здатність виявляти невідоме ШПЗ, системи виявлення аномалій часто характеризуються високим рівне хибних спрацювань завдяки високому.

Однією з можливостей, які шкідливе програмне забезпечення може гарантування невидиме для антивірусних засобів функціонування є повільна зміна поведінки в КС з метою приховування власної шкідливої діяльності.

Останнім часом створення методи виявлення ШПЗ на основі відслідковування їх поведінки стають стало альтернативними для його ефективного виявлення [6-10].

Пов'язані роботи

Питанню використання методів аналізу поведінки ПЗ в КС для захисту комп'ютерних систем від шкідливого ПЗ сьогодні присвячено велика кількість наукових робіт.

Зокрема, в [11] вперше було запропоновано описувати очікувану поведінку деяких привілейованих програм (наприклад, setuid root-програм, демонів в Unix, тощо) за допомогою мови опису програмної політики. Під час виконання програми будь-яке відхилення від зазначеної поведінки розглядалося як «зловживання». Основним обмеженням цього методу є складність опису очікуваної поведінки та написання специфікацій для всіх піддослідних програм. Тим не менш, це дослідження відкрило нові можливості для розроблення методів виявлення ШПЗ на основі описів поведінок.

В [6] була описана ідея використання коротких послідовностей системних викликів, виконаних запущеними програмами, як дискримінатор для виявлення ШПЗ. Нормальна поведінка визначалася в термінах коротких послідовностей системних викликів певної довжини під час виконання процесу в КС, і окрема база даних звичайної поведінки була побудована для кожного досліджуваного процесу. Ця робота була розширена різними схемами класифікації, такими як штучна імунна система [7], виведення правил [8] та прихована Марковська модель (HMM) [9]. В [10] застосовано методи штучної нейронної мережі для вивчення профілів поведінки програм із послідовностями системних викликів для даних DARPA BSM. Для кожної програми нейронна мережа була навчена та використана для виявлення аномалій. Їхні рекурентні мережі Елмана змогли виявити 77,3% всіх вторгнень без помилкових спрацьовувань і 100% всього ШПЗ із приблизно 10% помилкових спрацювань.

На відміну від більшості робіт, які були зосереджені на створенні профілів окремих програм, в [12] запропоновано метод, що базується на дискримінантному аналізі. Не вивчаючи всі системні виклики, класифікація процесу виконується шляхом аналізу лише 11 системних викликів у запущеній програмі та обчислення відстаней Махаланобіса до груп звичайних і шкідливих тренувальних процесів. Було зроблено 4 помилкових класифікації із 42 зразків. Проте, з огляду на невеликі розміри зразкових даних, все ще не встановлена ефективність такого підходу.

У [13] зроблено спробу порівняти ефективність методів виявлення ШПЗ, які використовували частоти системних викликів, та ті, що використовували порядок системних викликів. Імена системних викликів були вилучені із журналів як нормальних, так і шкідливих програм, і позначені як нормальні та шкідливі, відповідно. Щоправда, метод не передбачає групування системних викликів за програмами, що їх виконували. Оскільки як частота, так і упорядкованість системних викликів залежать від програми, це спрощення обмежує вплив їх роботи.

Із вище перерахованого можна зробити висновок, що на сьогоднішній день практично всі дослідження з вивчення поведінки програм використовують короткі послідовності системних викликів в якості ознак, та формують великі бази даних послідовностей системних викликів для кожної програми. Нормальна поведінка характеризується локальним порядком системних викликів, та відходження від нього розглядається як аномалія. Цей підхід вимагає значних накладних затрат і вимагає складного процесу створення нових профілів для кожної нової версії програми.

З огляду на це виникає задача розробки нового методу виявлення шкідливого ПЗ, що комбінує високу ефективність виявлення нового шкідливого ПЗ, низький рівень хибних спрацювань та низьку обчислювальну складність при використанні.

Метод виявлення шкідливого програмного забезпечення на основі алгоритму k-найближчих сусідів

В роботі представлено метод виявлення шкідливого програмного забезпечення на основі алгоритму k-найближчих сусідів, який здійснює класифікацію ПЗ на шкідливе і нормальне. Метод передбачає аналіз поведінки програмного забезпечення в комп'ютерній системі. В процесі роботи методу кожен досліджуваний процес представляється у вигляді вектора, у якому кожен елемент є значення статистичного показника, що відображає кількість системних викликів, які процес здійснює на протязі його виконання.

Метод дозволяє забезпечити реагування на нові загрози, забезпечуючи захист комп'ютерних систем від як відомого так і невідомого ШПЗ. Робота системи виявлення ШПЗ здійснюється на основі виявлення

аномалій через порівняння поведінки досліджуваного процесу із базою даних відомих поведінок програмного забезпечення в комп'ютерних системах.

Для опису поведінки програмного забезпечення в роботі пропонується розглядати системні виклики, які здійснюються ПЗ, не з точки зору локального порядку викликів, а з точки зору частоти викликів. Використовуючи апарату метафор обробки тексту, кожний системний виклик розглядатимемо як "слово" у довгому документі, а набір системних викликів, виконаних процесом, розглядатимемо як "документ". Ця аналогія дозволяє використовувати поний спектр засобів обробки тексту для задачі виявлення шкідливого ПЗ. В роботі з цією метою залучено одним із таких методів – алгоритм *k*-найближчих сусідів.

Використовуючи подібність у підходах до класифікації текстів, кожен процес можна представити у вигляді вектору, в якому кожен елемент представляє собою факт виконання певної кількості системних викликів під час його виконання. З метою визначення значень елементів вектору метод оперує частотне зважуванням та tf-idf. Для визначення процесу як нормальний або шкідливий класифікатор КНС здійснює обчислення схожості між даним процесом та кожним із, присутнім у навчальній вибірці та використовує класи k-найближчих унавчальних зразків для класифікації даного процесу. Основне припущення полягає в тому, що процеси, що належать одному класу, будуть групуватися разом у векторному просторі. Таблиця 1 ілюструє подібність деяких аспектів між класифікацією тексту та виявленням ШПЗ при застосуванні класифікатора КНС.

Є визначені переваги для застосування методів класифікації тексту до процесу виявлення ШПЗ. Перш за все, множина системних викликів є чітко обмеженою. Таким чином, розмірність векторів наперед відома і є визначеною. По-друге, процес виявлення ШПЗ може розглядатися як задача бінарної категоризації, що робить застосуванням методів класифікації тексту відносно виправданим, застосовним і простим в реалізації.

Таблиця 1

Аналогії між категоризацією текстів і програмним забезпеченням

Символ	Текст	ШПЗ
N	Кількість документів	Кількість процесів
M	Кількість слів	Кількість системних викликів
n_i	Кількість *i*-ого слова	Кількість *i*-ого виклику
f_{ij}	Частота *i*-ого слова в документі *j*	Частота *i*-ого виклику в процесі *j*
D_j	*j*-ий тренувальний документ	*j*-ий тренувальний процес
X	Тестовий документ	Тестовий процес

Таким чином, метод виявлення шкідливого програмного забезпечення на основі алгоритму *k*-найближчих сусідів включає в себе два етапи:

1. Навчання класифікатора КНС *k*-найближчих сусідів.
2. Використання класифікатора КНС *k*-найближчих сусідів для виявлення ШПЗ.

Розглянемо детальніше кроки першого етапу.

Отримання журналів виконання процесів на «чистій» системі в процесі її нормального функціонування. Вирішення задачі виявлення ШПЗ ґрунтується на детекції шкідливої поведінки виключно на основі порівняння з множиною нормальних поведінок ПЗ. Щоб забезпечити максимально покриваючу множину таких поведінок для виявлення аномалій необхідно використовувати великий набір тренувальних даних.

Отже, метод передбачає для забезпечення навчальної вибірки і для подальшого навчання КНС класифікатора використання так званої «чистої» комп'ютерної системи, на якій моделюється нормальна поведінка процесів. Для отримання журналів множини «нормальних» поведінок використовується служба трасування виконання процесів.

Навчання класифікатора. З метою здійснення процесу навчання класифікатора формується матриця *A,* в якій кожний елемент a_{ij} містить статистичний показник отриманий із кількісті виконань n_{ij} певного системного виклику *i* під час виконання процесу *j* (частота (1) чи tf-idf (2)):

$$a_{ij} = f_{ij} = \frac{n_{ij}}{\sum_{l=1}^{M} n_{ij}}, \tag{1}$$

$$a_{ij} = \frac{f_{ij}}{\sqrt{\sum_{l=1}^{M} f_{lj}^2}} \times \log\left(\frac{N}{n_i}\right), \tag{2}$$

де N – кількість навчальних поведінок (стовпців матриці), M – кількість системних викликів (рядки матриці).

Розглянемо детальніше кроки другого етапу.

Здійснення моніторингу подій в системі. Для кожного досліджуваного процесу формується вектор *X*, кожен із елементів якого є показник отриманий із кількісті виконань певного системного виклику *i* під час виконання процесу *j*. Для реалізації даного пункту необхідним є вирішення задачі трасування виконання процесів з метою отримання журналів поведінок програмного забезпечення в комп'ютерній системі в

реальному часі.

Використання алгоритму КНС для класифікації програмного забезпечення. Основним компонентом методу є здійснення класифікації програмного забезпечення на основі залучення алгоритму КНС. Ця задача зводиться до оцінки подібності між вектором X та кожним стовпцем у матриці A. З цією метою використовується косинусна подібність (3) Якщо середня подібність до k найбільш подібних векторів в A перевищує певне порогове значення, досліджуваний процес класифікується як нормальний, інакше – як шкідливий.

$$sim\left(X, D_j\right) = \frac{\sum_{t_i \in \left(X \cap D_j\right)} x_i \times d_{ij}}{\left\|X\right\|_2 \times \left\|D_j\right\|_2}, \tag{3}$$

де X – досліджуваний процес, D_j – j-ий тренувальний процес, t_i – спільний виклик між X та D_j, x_i – вага виклику t_i в X, d_{ij} – вага виклику t_i в D_j, $||X||_2$ – довжина вектору X; $||D||_2$ – довжина вектору D_j.

Якщо оцінка подібності із одним тренувальним нормального процесом дорівнює 1, це означає, що частота системних викликів досліджуваного та навчального процесів відповідають повністю, і досліджуваний процес буде класифікований як «нормальний» процес. В іншому випадку навчальні поведінки сортуються за значеннями подібності за спаданням, і вибирається k поведінок із найбільшою подібністю, щоб визначити, чи є процес нормальним чи шкідливим. Потім здійснюється обчислення середнього значення подібності k найближчих сусідів (з найбільшими оцінками подібності) і встановлюється порогове значення. Тільки тоді, коли середнє значення подібності перевищує порогове значення, новий процес вважається нормальним. Псевдокод для алгоритму КНС представлений у вигляді наступного алгоритму:

```
побудувати тренувальний набір нормальних процесів D;
for кожний процес X в тестовому наборі даних do
  if X містить невідомий виклик then
    X – ненормальний;
  else
    for кожний процес Dj в D do
      обрахувати sim(X,Dj);
      if sim(X,Dj)=1 then
        X – нормальний; exit;
      Знайти sim_avg – середнє значення k найбільших sim(X,Dj);
      if sim_avg>поріг then
        X – нормальний
      else
        X – ненормальний
```

Прийняття заходів на основі результатів роботи КНС. Запропонований метод не здатний виявити ШПЗ до його виконання. Проте, функціонування ШПЗ складається з багатьох етапів, тому блокування в процесі виконання шкідливого ПЗ є ефективним заходом в захист КС. У випадку класифікації процесу як шкідливого метод передбачає його повне блокування та відправку на аналіз адміністратору мережі., а також його подальше внесення в «чорний список», що завадить її повторному виконанню.

Експериментальні дослідження

З метою дослідження ефективності запропонованого методу було проведено ряд експериментів.

Перша частина експериментів полягала в зборі та опрацювання поведінки «чистих» комп'ютерних систем з метою навчання класифікатора КНС, для чого був використаний набір навчальних даних DARPA IDSE [14]

На другому етапі здійснювалося безпосередня оцінка ефективності розробленої системи. З цією метою було згенеровано множину шкідливих поведінок в комп'ютерній системі та використано дані DARPA, що являють собою велику вибірку комп'ютерних атак, вбудованих у звичайний фоновий трафік, які використовує ШПЗ. Серед фонових даних містилися ШПЗ, що здійснювало 28 типів мережевих атак та 12 сценаріїв шкідливої локальної активності в КС.

В процесі виконання експериментів було використано дані, зібрані з досліджуваної комп'ютерної системи, що функціонує в локальній мережі. Журнали містили інформацію про стан комп'ютерної системи під час її роботи. Із цих журналів виділялися лише назви системних викликів для кожного процесу. Інші атрибути подій, такі як аргументи системних викликів, шляхи та атрибути об'єктів, повернені значення тощо, не використовувались.

Дані DARPA було позначені номерами сесій. Кожна сесія відповідала TCP/IP-з'єднанню між двома комп'ютерними системами, що брали участь в експериментах.

Кожна сесія складалася з одного або декількох процесів. Для кожного процесу згенерований повний впорядкований список системних викликів.

Були підраховані кількості окремих системних викликів під час виконання процесу. Потім із цих кількостей було сформовано вектори ознак процесів. Для кодування процесів було використано два показники, відносні частоти викликів (1) та tf-idf (2).

На протязі тренування було спостережено 50 різних системних викликів, що означає що розмірність кожного вектору процесу має бути 50.

Після того, як було отримано навчальні дані про множину нормальних поведінок КС в комп'ютерній мережі, запропонований метод виявлення ШПЗ було застосовано до класифікації нового ШПЗ.

Результати ефективності роботи методу у подано у вигляді ROC-кривих. Зокрема, в результаті експериментальних досліджень було отримані різні співвідношення між рівнями виявлення ШПЗ та хибних спрацювань. З цією метою було досліджено значення порогу, при якому досягалася найвища ефективність методу.

Ефективність алгоритму класифікатора КНС також залежить від значення k, числа найближчих сусідів трестованого процесу. Зазвичай оптимальне значення k визначається емпірично. З цією метою було проведено ряд експериментів зі зміною величини k від 5 до 25. На рисунку 2 показані криві ROC для 3 різних k значень, коли в якості ознак було використано показник tf-idf.

Результати досліджень продемонстрували, що для досягнення найвищих показників виявлення значення k = 10 є кращим вибором, ніж інші значення, оскільки відсоток виявлення досягає максимального значення (98%) швидко, коли пороговий рівень піднімається до 0,72, а відсоток хибних спрацювань становить лише 3.04% для всього набору тестувальних даних.

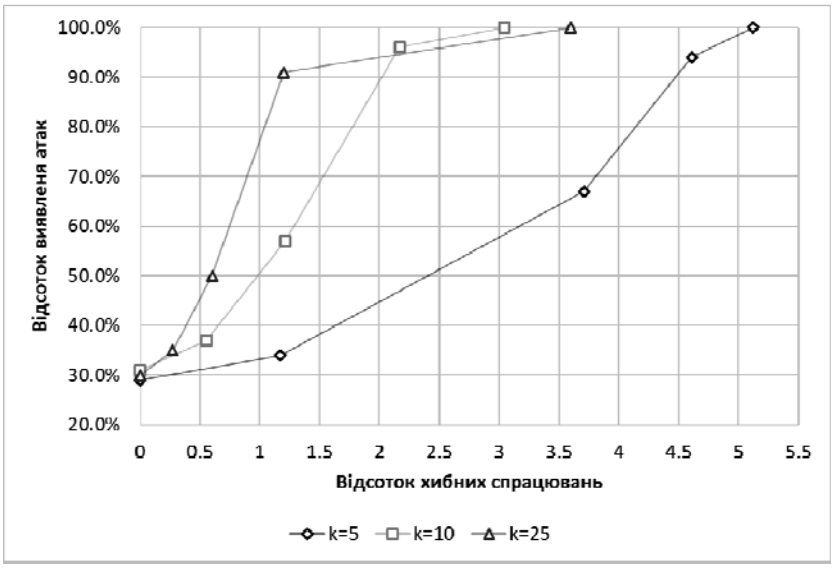

Рис. 1. Результати роботи методу при різних значеннях k

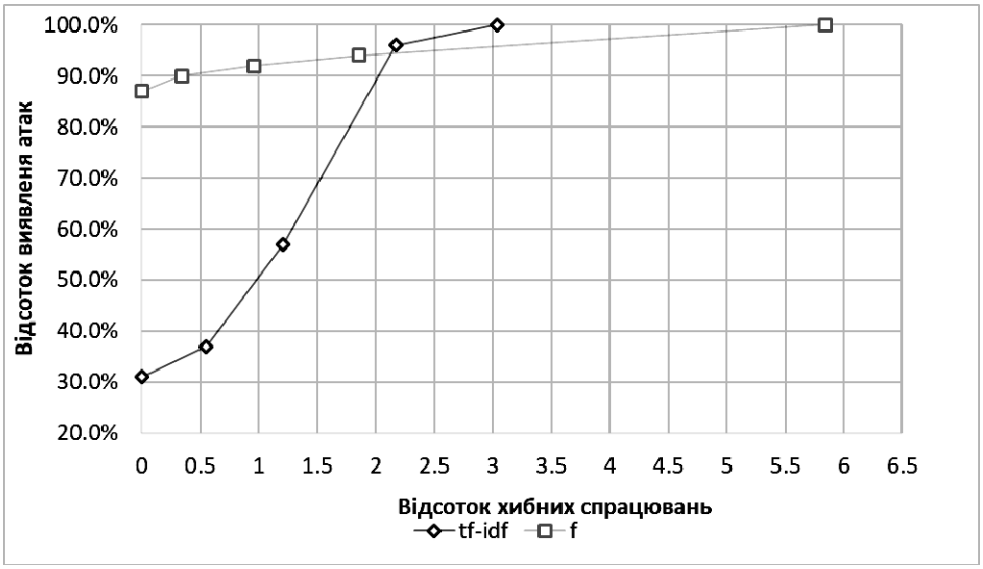

Рис. 2. Порівняння класифікаторів із використанням частоти та tf-idf

Експерименти також включали дослідження застосування відносної частоти системних викликів в якості ознак. Аналогічним чином, для методу частотного зважування, k = 15 забезпечує найнижчий відсоток хибних спрацювань 5,84% коли відсоток виявлених ШПЗ досягає 96% при пороговому значенні 0,99. Причиною високого порогового значення є те, що деякі поведінки ШПЗ дуже схожі на звичайні процеси при коли використовуються відносні частоти в якості ознак зважуванні. Результати порівняння двох різних методів показано на рисунку 3. У той час як метод із використанням частоти демонструє високий відсоток виявлення ШПЗ (87%) при найнижчому рівні помилкових спрацьовувань, метод що використовує tf-idf

забезпечує нижчий відсоток хибних спрацювань при високому рівні виявлення ШПЗ. Звідси видно, що використання tf-idf дозволяє краще розрізняти процеси, ніж частоти. Отже, tf-idf вимагає нижче порогове значення і надає нижчий відсоток хибних спрацювань.

Таким чином, розроблений метод виявлення шкідливого програмного забезпечення на основі методу *k*-найближчих сусідів дає змогу здійснювати виявлення ШПЗ з високою достовірністю.

Висновки

В роботі представлено метод виявлення шкідливого програмного забезпечення на основі алгоритму k-найближчих сусідів, який здійснює класифікацію ПЗ на шкідливе і нормальне. Метод передбачає аналіз поведінки програмного забезпечення в комп'ютерній системі. В процесі роботи методу кожен досліджуваний процес представляється у вигляді вектора, у якому кожен елемент є значення статистичного показника, що відображає кількість системних викликів, які процес здійснює на протязі його виконання.

Метод дозволяє забезпечити реагування на нові загрози, забезпечуючи захист комп'ютерних систем від як відомого так і невідомого ШПЗ. Робота системи виявлення ШПЗ здійснюється на основі виявлення аномалій через порівняння поведінки досліджуваного процесу із базою даних відомих поведінок програмного забезпечення в комп'ютерних системах.

Експериментальні дослідження запропонованого методу показали застосовність методу та можливість його програмної реалізації у вигляді програмного забезпечення на рівні 96-98%, при 3-6% хибних спрацювань.

Для оцінки та виокремлення характерних для ШПЗ поведінок метод застосовує статистичні показники tf-idf, що дозволяє отримати оптимальне співвідношення між рівнем виявлення та рівнем хибних спрацювань. Експериментальні дослідження показали необхідні статистичні показники та їх параметри для досягнення високої ефективності виявлення ШПЗ при низькому рівні хибних спрацювань.

На основі виявлених шкідливих процесів можна автоматично побудувати сигнатури для більш ранніх методів виявлення ШПЗ.

Недоліком методу є нездатність виявлення ШПЗ до того початку його запуску на виконання.

References

1. Xiaoyong Yuan "Deep Learing-based Real-Time Malware Detection with Multi-Stage Anaysis" SMARTCOMP 2017

2. S. Kumar and E. H. Spafford, "An application of pattern matching in intrusion detection," 1994

3. M. Christodorescu, S. Jha, and C. Kruegel, "Mining specifications of malicious behavior," in Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, ser. ESEC-FSE '07, 2007, pp. 5–14.

4. V. N. P. Dao and V. R. Vemuri, "Computer Network Intrusion Detection: A Comparison of Neural Networks Methods", Differential Equations and Dynamical Systems, (Special Issue on Neural Networks, Part-2), vol.10, No. 1&2, 2002.

5. Shabtai A, Moskovitch R, Elovici Y, Glezer C: Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. Information Security Technical Report 2009, 14(1):1-34.

6. S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Logstaff, "A Sense of Self for Unix process", Proceedings of 1996 IEEE Symposium on Computer Security and Privacy , 120- 128, 1996

7. S. Forrest, S. A. Hofmeyr and A. Somayaji, "Computer Immunology", Communications of the ACM , Vol. 40, 88-96, 1997.

8. W. Lee, S. J. Stolfo and P. K. Chan, "Learning Patterns from Unix Process Execution Traces for Intrusion Detection", Proceedings of AAAI97 Workshop on AI Methods in Fraud and Risk Management , 50-56, 1997.

9. C. Warrender, S. Forrest and B. Pearlmutter, "Detecting Intrusions Using System Calls: Alternative Data Models", Proceedings of 1999 IEEE Symposium on Security and Privacy , 133-145, 1999.

10. A. K. Ghosh, A. Schwartzbard and A. M. Shatz, "Learning Program Behavior Profiles for Intrusion Detection", Proceedings of 1st USENIX Workshop on Intrusion Detection and Network Monitoring , Santa Clara, CA, April 1999.

11. C. Ko, G. Fink and K. Levitt, "Automated Detection of Vulnerabilities in Privileged Programs by Execution Monitoring", Proceedings of 10th Annual Computer Security Applications Conference , Orlando, FL, Dec, 134-144, 1994.

12. M. Asaka, T. Onabuta, T. Inoue, S. Okazawa and S. Goto, "A New Intrusion Detection Method Based on Discriminant Analysis", IEEE TRANS. INF. & SYST ., Vol. E84-D, No. 5, 570-577, 2001.

13. N. Ye, X. Li, Q. Chen S. M. Emran and M. Xu, "Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data", IEEE Trans. SMC-A , Vol. 31, No. 4, 266-274, 2001.

14. MIT Lincoln Laboratory, https://www.ll.mit.edu/ideval

Рецензія/Peer review : 11.11.2017 р. Надрукована/Printed : 01.12.2017 р.
Рецензент: д.т.н., проф. Боровик О.В.