УДК 004.056.5
DOI 10.31891/2307-5732-2020-285-3-7

С. М. ЛИСЕНКО, В. Ю. ОМЕЛЬЯНЕНКО, Р. В. ЩУКА
Хмельницький національний університет

МЕТОД ІДЕНТИФІКАЦІЇ ШПИГУНСЬКОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В КОМП'ЮТЕРНИХ СИСТЕМАХ

У роботі представлено подальший розвиток методу ідентифікації шпигунського програмного забезпечення в комп'ютерних системах, який дозволяє виявляти усі типи, і відрізняється від відомих тим, що забезпечує принцип проактивності та базується на механізмах машинного навчання з підкріпленням. Запропонований метод ідентифікації шпигунського програмного забезпечення використовує аналіз поведінки програмного забезпечення в комп'ютерних системах. Метод ґрунтується також на аналізі даних із залучення апарату машинного навчання, зокрема навчання з підкріпленням, який оперує з поняттями значень стабільності для виявлення ШПЗ. Запропонований метод використовує рівняння отримання нової інформації, здійснює обчислення величини впливу кожної з властивостей програмного забезпечення в потрібній категорії, а завдяки ефективності стабільності EAX, значення присвоюється до нього в якості нагороди.

Ключові слова: шкідливе програмне забезпечення, шпигунське програмне забезпечення, ідентифікація, машинне навчання, навчання з підкріпленням, комп'ютерна система.

S. LYSENKO, V. OMELIANENKO, R. SHCHUKA
Khmelnytskyi National University

METHOD FOR THE SPYWARE IDENTIFICATION IN THE COMPUTER SYSTEMS

The paper presents a method of identification of spyware identification software in computer systems, which allows detection of all types, and differs from the known ones, which provides the principle of proactivity and is based on mechanisms of machine learning. For the proposed method of identifying spyware, it uses an analysis of the behaviour of the software on computer systems. The method is also based on a downstream analysis of the involvement of machine learning apparatus, in particular reinforcement learning, which operates with concepts of stability values for the malware detection. The proposed method, using the equation for obtaining new information, calculates the value of the impact of each of the properties of the software in the desired category, and due to the efficiency of EAX stability, the value is assigned to it as a reward. Here are some basic steps in the method of identifying spyware on computer systems: tracking the behaviour of executable files on a computer system; calculation of stability values; selection of features that may indicate the presence of spyware on computer systems by equating the retrieval of new information; calculating the reward value based on the EAX stability value; building a spyware identification template; software instance analysis; calculating the stability value of the software instance under study; adaptation of a file of a file with the obtained stability values for comparison with malicious and useful instances of files based on the stability values of known spyware. Method for the spyware identification in the computer systems enables the detection of the URL logger, Keyloggers, Screen Recorders, Chat loggers, Email logger, Password logger, Password hijackers, Keyloggers and password loggers, logger, Advertising Programs (Web Errors), Browser Hijacking, Modem abduction, PC abduction, Information thefts (*Infostealers*) *and Banking Trojans. Proposed method demonstrated the possibility of spyware detection with high reliability up to 97.55 %.*

Keywords: malware, spyware, machine learning, identification, reinforcement learning, computer system.

Вступ. Шпигунське програмне забезпечення – це тип шкідливого програмного забезпечення, яке проникає в операційну систему (ОС) комп'ютерної системи (КС) зловмисним способом [1]. Вони розроблені для відстеження дій користувачів у КС та Інтернеті, викрадаючи конфіденційну інформацію [2].

За підрахунками, понад 80 % усіх КС інфіковано певним шпигунським програмним забезпеченням [3, 4].

Існує помилкова думка, що шпигунське програмне забезпечення – це комп'ютерний вірус. Комп'ютерний вірус – це фрагмент коду, призначений для реплікації себе якомога більше разів, поширюючись з одного хост-комп'ютера на будь-який інший комп'ютер, підключений до нього.

Шпигунське ПЗ не призначене для пошкодження КС. Шпигунське програмне забезпечення визначається загалом як будь-яка програма, яка потрапляє в КС без дозволу користувача і приховує своє функціонування у фоновому режимі, в той час як вносить небажані зміни у функціонал ОС. Шкода, яку завдає ШПЗ полягає у тому, щоб перетворити користувача на ціль реклами або змусити браузер користувача відображати певні сайти чи результати пошуку [3].

Відомі методи ідентифікації шпигунського програмного забезпечення демонструють невисоку достовірність проактивного виявлення, тому актуальною задачею є розроблення нових методів, що вирішують вище вказану задачу забезпечення інформаційної безпеки КС.

Пов'язані роботи. Існує багато сучасних методів та підходів до ідентифікації шпигунського програмного забезпечення.

В [5] запропоновано метод аналізу поведінки ПЗ, в якій послідовності системних викликів отримують та представляють інструкціями MIST. Тоді за допомогою методу N-грам виконується генерація набору функцій на основі зібраних даних системного виклику. В [6] представлено метод ідентифікації ШПЗ, який дозволяє виявляти системні виклики шляхом моніторингу та їх перетворення у впорядковані граф системних викликів (OSCG). У [7–11]] виявлення здійснюється за допомогою аналізу частотних викликів системних функцій, а також залучалася кластеризація зразків зловмисного програмного забезпечення в групи, а на основі методу найближчого спільного сусіда (SNN). В [12] представлено результати вилучення послідовності опкодів кожного блоку відповідно до структури потоку управління зразками шкідливих програм, а потім обчислено хеш-функцію для формування множини функцій. Потім функції були обрані за

допомогою IDF та використовувались для пошуку правил класифікації шкідливих та корисних зразків. В [13–15] надано граф потоку зразків шкідливих програм у вигляді дерева виконання, щоб отримати шляхи виконання. Вони об'єднали всі можливі шляхи для формування потоку опкодів і використовували метод N-грам для отримання функцій поведінки. Крім того, IG та DF використовувались для вибору особливостей поведінки. В [16] запропоновано метод, який отримує ознаки про виклики з коду розбирання зразків шкідливого програмного забезпечення. Використовуючи статистичні методи, частоти викликів API підраховуються і виконується висновок щодо присутності ШПЗ на основі методів IG та CHI. В [17–19] запропонувала IT, яка дозволяє виконати класифікацію шкідливих програм. Система включає в себе бази даних графів з API викликів та створює графічні вектори на основі графів, які несуть ненульовий показник подібності в одному елементі лише у тому випадку, якщо відповідний граф найкраще відповідає одному з графів.

Таким чином, аналіз ефективності методів [5–19] продемонстрував їх невисоку достовірність ідентифікації шпигунського програмного забезпечення, а також високий рівень хибних спрацювань.

Метод ідентифікації шпигунського програмного забезпечення. В роботі представлено подальший розвитку методу ідентифікації шпигунського програмного забезпечення в комп'ютерних системах, який дозволяє виявляти усі типи, і відрізняється від відомих тим, що забезпечує принцип проактивності та базується на механізмах машинного навчання з підкріпленням.

Запропонований метод ідентифікації шпигунського програмного забезпечення використовує аналіз поведінки програмного забезпечення в комп'ютерних системах.

Метод ґрунтується також на аналізі даних із залучення апарату машинного навчання, зокрема навчання з підкріпленням, який оперує з поняттями значень стабільності для виявлення ШПЗ.

Запропонований метод, використовує рівняння отримання нової інформації, здійснює обчислення величини впливу кожної з властивостей програмного забезпечення в потрібній категорії, а завдяки ефективності стабільності EAX, значення присвоюється до нього в якості нагороди.

Розглянемо основні кроки методу ідентифікації шпигунського програмного забезпечення в комп'ютерних системах:

1. Відстеження поведінки виконуваних файлів в комп'ютерній системі.
2. Обчислення значення стабільності.
3. Відбір ознак, що можуть свідчити про присутність шпигунського програмного забезпечення в комп'ютерних системах за допомогою рівняння отримання нової інформації.
4. Обчислення значення нагороди на основі значення стабільності EAX.
5. Побудова шаблону ідентифікації шпигунського програмного забезпечення.
6. Аналіз екземпляру ПЗ.
7. Обчислення значення стабільності досліджуваного екземпляру ПЗ.
8. Адаптація екземпляру файлу з отриманими значеннями стабільності для порівняння з шкідливими і корисними екземплярами файлів на основі значень стабільності відомого шпигунського програмного забезпечення.

Розглянемо детальніше кроки методу.

Збір інформації. Згідно з термінологією машинного навчання з підсиленням поняття необхідним є обчислення значення винагороди. З цією метою необхідним є використання рівняння приросту інформації. Беручи до уваги, що в цьому рівнянні співвідношення впливу кожного з атрибутів вказано відповідно до результату, отримане значення за допомогою цього рівняння являє собою значення між нулем і одиницею [20]. Чим ближче це значення до числа 1, тим більше впливу має функція на отримання правильної відповіді.

Приймемо, що якщо pi – це ймовірність того, що множник даних D належить до класу C_i, значення C_i оцінюється $|C_i, D| / |D|$. Очікується, що необхідна інформація щодо множини в D обчислюється за допомогою рівняння [20]:

$$\mathrm{Info}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i). \quad (1)$$

Дані потребують класифікації на D (після використання A для поділу D на V секції). Тоді класифікація об'єктів дослідження здійснюватиметься за формулою: [21]:

$$\mathrm{Info}_A(D) = \sum_{j=1}^{y} \frac{|D_j|}{|D|} \times I(D_j). \quad (2)$$

Функція отримання нової інформації – це різниця між початковими необхідними даними та новими необхідними даними.

Іншими словами, отримання (A) означає ступінь впливу, на який впливає збільшення функції A [22]:

$$\mathrm{Gain}(A) = \mathrm{Info}_A(D). \quad (3)$$

Застосування навчання з підкріпленням для ідентифікації шпигунського програмного забезпечення. Механізм навчання з підкріпленням полягає в тому, щоб дізнатися, як стан досліджуваного об'єкту відображається в дію, коли є тільки один прояв.

У процесі навчання з підкріпленням системи для ідентифікації шпигунського програмного забезпечення здійснюється дослідження фактора, що зазнає випробувань та обчислення помилок для вивчити оптимальні дії, які слід здійснити в кожному стані досліджуваного об'єкта та для досягнення мети.

Згідно теорії машинного навчання з підкріпленням, фактор вибирає дію у будь-якому випадку, в якому, можливо, що агент отримує новий стан або винагороду [23].

Повторюючи цей процес, агент дізнається, що найкраща дія для отримання максимального очікування – це накопичення винагород. Загалом, в кожному повторенні агент змінює розуміння поточних станів s, s∈S, вибір дії a, a∈A, і, ймовірно, його стан і отримує нагороди r, r∈R.

У цьому процесі, щоб отримати корисний досвід щодо стану, агента, перехід стану і винагороди, агенту потрібні оптимальні дії та оцінка системи, що відповідає процесу навчання [23].

На рис. 1 наведений процес навчання системи згідно теорії машинного навчання з підкріпленням.

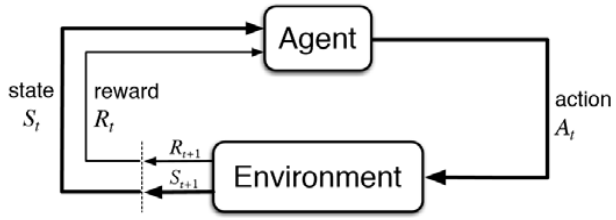

Рис. 1. Процес навчання з підсиленням

У запропонованому методі ідентифікації шпигунського програмного забезпечення застосовано метод навчання з підкріпленням – алгоритм часової різниці.

Алгоритм часової різниці має суттєві переваги щодо навчання, оскільки не вимагає попереднього вивчення поведінки шпигунського програмного забезпечення, має високі можливості в покроковому обчисленні адитивності й ідентифікації [15]. У цьому методі агент намагається оцінити функцію значення, виходячи з моменту нарахування моменту та оціненої значення винагороди для наступного стану досліджуваного об’єкта – найпростішим і найбільш корисним часовим різницевим методом навчання – методом Q навчання.

Метод Q навчання використовує досвід, отриманий від будь-якого переходу стану в середовищі, і оновлює значення, пов'язані з кожним станом, в таблиці під назвою Q.

Така таблиця містить стан і дію Q(S, a) для кожної пари та для кожного стану перехід від S_t до S_{t+1} та отримання нагороди r_{t+1}, оновлює кількість Q у таблиці відповідно до рівняння:

$$Q(S_t, a_t) = Q(S_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(S_{t+1}, a_t) - Q(S_t, a_t)]. \quad (4)$$

Коли агент вибирає дію a у стані S, а потім переходить у стан S', використовує максимальне значення Q наступного стану, тобто максимальне Q(s', a') для оновлення значення Q в цей момент, тобто Q(S, a). У цьому алгоритмі, якщо будь-яка дія a в будь-якому випадку, вибирається нескінченний час у нескінченному запуску програми, а також якщо параметр α добре відрегульований, значення Q зблизиться з Q* з ймовірністю a 100 %. У запропонованому методі ідентифікації шпигунського програмного забезпечення використовується метод тимчасових різниць, який застосовує лише зворотний зв'язок щодо отримання винагороди за краще навчання та виявлення ШПЗ.

Таким чином, метою вибору навчання з підкріпленням для вирішення проблеми ідентифікації шпигунського програмного забезпечення є використання зворотного зв’язку щодо отримання винагороди, яка може краще виражати більш впливові дані та призводить до виявлення з високою точністю.

Аналіз стану досліджуваного об’єкта. Стан відноситься до умов, що впливають на прийняття рішень агента в навколишньому середовищі, такі як місце розташування агента в навколишньому середовищі.

Запропонований метод оперує двома стани досліджуваного об’єкта для ідентифікації:

1. S1: Коли значення стабільності регістра EAX змінюються на 2 після виклику.
2. S2: Коли значення стабільність регістра EAX не змінюється на 2 після виклику.

Необхідно вказати область застосування двох станів. $S = \{S_i, | i = 1, 2\}$.

Колекція передбачуваного стану має властивості ланцюгів Маркова. Якщо стан має властивість Маркова, це означає, що він містить в собі поточний стан всієї інформації, пов'язаної з минулим і сучасним, які необхідні для продовження навчання.

Аналіз дій досліджуваного об’єкта. Система ідентифікації ШПЗ виконує такі дії:

1. Стабілізація змін, що призводять до створення шпигунського програмного забезпечення.
2. Стабілізація змін, які не призводять до створення шпигунського програмного забезпечення.

Нагорода і покарання. Нагорода і покарання – це зворотний зв'язок, який отримав агент з навколишнього середовища, і є реакцією середовища, пов’язана з дією системи на нього.

Цей зворотний зв'язок буде реалізуватися наступним чином для системи ідентифікації ШПЗ:

1. Коли значення функції приросту більше, ніж середнє значення, вона буде отримувати винагороду по відношенню до певної формули для отримання винагороди.
2. Коли сума функції приросту нижче, ніж середнє значення, вона не буде отримувати будь-які винагороди, які розглядається в якості покарання.

Функція цінності. Функція цінності являє собою суму оптимальності дії в стані, в порівнянні з іншими діями в інших станах. В результаті, ця функція також залежить як від стану, так і від дії. Наприклад, Q(s, a) показана $Q(s_1, a_2)$ від суми оптимальності a_2 дії в стані s_1. Функція цінності може допомогти нам вибрати більш оптимальні дії в поточному стані. У процесі навчання, більш оптимальні дії набудуть більш високу якість у порівнянні з іншими діями.

Застосування алгоритмів навчання часової різниці. Запропонований метод оперує двома основними методами реалізації алгоритмів навчання: алгоритм Q-навчання та алгоритм SARSA.

Алгоритм Q-навчання часто має більш оптимальне прийняття рішень в порівнянні з алгоритмом SARSA, в той час як SARSA має більш стабільне прийняття рішень, а також може отримати більше нагород.

Розглянемо етапи алгоритму Q-навчання:

1. Значення Q(a, s) вважається рівним нулю для всіх станів і дій.
2. Отримуємо поточний стан системи.
3. Вибираємо дію на основі алгоритму.
4. Виконуємо дію в навколишньому середовищі, а потім чекаємо нагороди нашої дії.
5. Отримуємо новий стан системи (S'), до якої система переходить після виконання дії.

Оновлюємо кількість Q(a, s) на основі рівняння:

$$Q(S_t, a_t) = Q(S_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(S_{t+1}, a_t) - Q(S_t, a_t)], \quad (5)$$

де α – значення від нуля до одиниці, яке називається швидкістю навчання, і визначає швидкість навчання.

Очевидно, що чим більше значення α , тим більша швидкість навчання; Проте треба сказати, що велика кількість α робить навчання нестійким.

У більшості додатків величина 0,1, як правило, рекомендується для швидкості навчання, і ми використовували це значення. γ – це значення між 0 і 1, яке називається коефіцієнтом дисконтування, і він перешкоджає розбіжності функції якості під час навчання.

Значення γ зазвичай вважається рівним 0,9. У нашому методі, оскільки ми використовували лише величину винагороди, коефіцієнт дисконтування вважався рівним нулю.

Q_{max} (s', a') – це якість найбільш оптимальної дії в новому стані системи, а саме S′. Іншими словами, це величина максимальної якості в s' стані. Нарешті, перевіряється, чи досягає агент своєї мети чи ні (тобто, чи це зловмисне програмне забезпечення чи ні). Якщо відповідь була негативною (тобто це було зловмисне програмне забезпечення), алгоритм повторюється на третьому етапі на основі стану S'. В іншому випадку (тобто програмне забезпечення є доброякісним) алгоритм припиняється.

Експериментальні дослідження методу ідентифікації шпигунського програного забезпечення. З метою проведення експериментальних досліджень було згруповано за категоріями усі типи досліджуваного ШПЗ. Набір даних включає в себе 551 зразок ШПЗ [24].

Значення кількості зразків шпигунського ПЗ, задіяних експериментах представлено в таблиці 1.

Зібрані дані представляють собою файли, у яких кожен двійковий файл запускається в контрольованому середовищі, а також звіти під час виконання записуються на основі системи інтерфейсу прикладного програмування (API).

Інструмент WinAPI32 використовується для реєстрації виклику програмного забезпечення всіх бінарних файлів API, що містять вміст стабільності і повертається значення API.

Кожен файл працює протягом визначеного інтервалу часу. Цей інтервал вважається дві хвилини, що триває дві хвилини максимум, або час його виконання закінчується до завершення двох хвилин.

Причина вибору двохвилинного інтервалу є те, що шкідливі програми мають можливість запуститись, і навіть якщо вони мають корисне навантаження, вони також працюють.

WinAPI32 повідомляє про стабільність до і після виклику кожного бінарного файлу.

Однак, виконуваний файл не має можливості отримати зміст стабільності і пам'яті. Отже, звільнення і зміна значень деяких з стабільності досягаються в бінарному режимі.

Таблиця 1

Значення кількості зразків шпигунського ПЗ, задіяних експериментах

№ з/п	Тип шпигунського ПЗ	Значення кількості зразків, задіяних експериментах
1	Інтернет-реєстратори URL-адрес	37
2	Кейлогери	44
3	Екранні реєстратори	23
4	Чат реєстратори	45
5	Реєстратори електронної пошти	66
6	Реєстратори паролів	43
7	Викрадачі паролів	33
8	Кейлогери та реєстратори паролів	42
9	Рекламні програми (веб-помилки)	29
10	Браузер викрадення	54
11	Модем викрадення	14
12	ПК викрадення	34
13	Інформаційні крадіжки (Infostealers)	55
14	Банківські трояни	32

Зібрані дані представляють собою файли, у яких кожен двійковий файл запускається в контрольованому середовищі, а також звіти під час виконання записуються на основі системи інтерфейсу прикладного програмування (API).

Інструмент WinAPI32 використовується для реєстрації виклику програмного забезпечення всіх бінарних файлів API, що містять вміст стабільності і повертається значення API.

Кожен файл працює протягом визначеного інтервалу часу. Цей інтервал вважається дві хвилини, що триває дві хвилини максимум, або час його виконання закінчується до завершення двох хвилин.

Причина вибору двохвилинного інтервалу є те, що шкідливі програми мають можливість запуститись, і навіть якщо вони мають корисне навантаження, вони також працюють.

WinAPI32 повідомляє про стабільність до і після виклику кожного бінарного файлу.

Однак, виконуваний файл не має можливості отримати зміст стабільності і пам'яті. Отже, звільнення і зміна значень деяких з стабільності досягаються в бінарному режимі.

Витягуються тільки значення з колекції стабільності, які пов'язані з розрахунками і логічними операціями. Стабільність, яка передає дані або маніпулює стеком, не може бути реалізована, поки їх стабільність буде готувати середовище для обчислень, а їх значення можуть бути найменш репрезентативними для різних форм поведінки програмного забезпечення. Для того, щоб проаналізувати корисні та шкідливі файли потрібно, щоб файл був доступний на шляху його аналізу.

Щоб застосувати запропонований метод, на основі алгоритму спочатку необхідним є класифікація непроаналізованих попередньо даних. З цією метою для кожного з типу ШПЗ присвоюється число. Цей процес присвоєння значень здійснюється відповідно до таблиці 2. Слід зазначити, по відношенню до частоти чисел в кожній категорії, інтервали вказані, як показані в цій таблиці.

На наступному етапі, значення приросту інформації для кожної стабільності окремо отримують в усіх категоріях. Після отримання значень приросту інформації для всіх категорій, отримують середнє значення приросту інформації для кожної категорії.

Завдяки значень приросту інформації для кожної стабільності відносно середнього показника тієї ж категорії, стабільність EAX, яка вище середнього, вибирається в якості винагороди.

Таблиця 2

Класифікація значення регістрів процесора

Інтервал	Категорія
номер 0	0
номер 1	1
номер 2	2
[3 ... 9]	3
[10 ... 99]	4
[100 ... 999]	5
[1000 ... 99999]	6
[100000 ... 999999]	7
[AFFFFF ... A00000]	8
[BFFFFF ... B00000]	9
[CFFFFF ... C00000]	10
[DFFFFF ... D00000]	11
[EFFFFF ... E00000]	12
[FFFFFF ... F00000]	13
[FFFFFFFF ... A000000]	14

Таблиця 3 показує приклади середнє значення приросту інформації для типів ШПЗ.

Таблиця 3

Середнє значення приросту інформації кожної категорії

№ з/п	Категорія	Сума приросту інформації	Середнє значення
1	Інтернет-реєстратори URL-адрес	2,0778	0,0865
2	Кейлогери	0,7029	0,1126
3	Екранні реєстратори	2,4265	0,1011
4	Чат реєстратори	4,02837	0,1789
5	Реєстратори електронної пошти	4,3256	0,1802
6	Реєстратори паролів	2,6083	0,1086
7	Викрадачі паролів	4,41201	0,1838
8	Кейлогери та реєстратори паролів	4,3088	0,2512
9	Корисні файли	5,3088	0,2212

Після отримання середнього значення приросту інформації, стабільність EAX, яка отримує винагороду, буде визначатися в кожній категорії; і значення винагороди застосовується відповідно до пропонованого рівнянням.

Обчислення винагород: Нагорода = Середнє значення приросту інформації × 0.5 × приросту інформації стабільності EAX.

Причина вибору числа 0,5 у пропонованій формулі полягає в тому, що приріст інформації є значенням від 0 до 1, і є необхідність застосування коефіцієнту так, щоб отримане значення винагороди стало нормальним і збалансованим по відношенню до всіх даних. Наприклад, значення винагороди за стабільність EAX в категорії вірусів розраховується за формулою (5) наступним чином:

Нагорода стабільності EAX = 0,365×0,5×0,0865, приріст інформації EAX = 0,365, середнє значення приросту інформації категорії ШПЗ = 0.0865, а також, фактор винагороди = 0,5 і, отже: нагорода стабільності EAX = 0,0157.

Для інших груп значення отримують таким же чином, і значення винагороди отримують для цієї стабільності. Після отримання значення винагороди за кожної з даних окремо у всіх категоріях, відповідно до таблиці 5, застосовується нове значення для забезпечення стабільності EAX з використанням навчання з підкріпленням. У таблиці 4 значення нагороди за всі категорій окремо обчислюється, і значення винагороди кожної категорії та пов'язані з нею номери розташовані в рівнянні. Отримане значення замінює попереднє значення в стабільності EAX. У цій таблиці, тільки одне значення відображається в кожній категорії, а решта цифри перевіряються таким же чином. Тепер, після застосування цих змін, дані готові для аналізу.

Таблиця 4

Нове значення регістра EAX в кожній категорії ШПЗ, використовуючи навчання з підкріпленням

Тип ШПЗ (категорія)	Початкове значення Q (стабільність)	Нове значення Q (нове значення стабільності EAX)
Інтернет-реєстратори URL-адрес	7	6,30157
Кейлогери	12	10,80237
Екранні реєстратори	8	7,20208
Чат реєстратори	9	8,10572
Реєстратори електронної пошти	11	9,90505
Реєстратори паролів	13	11,70231
Викрадачі паролів	10	9,00608
Корисні файли	14	12,60704

Для оцінки ефективності роботи запропонованого методу скористаємося параметрами:

1. TP (true positives) – кількість шкідливих поведінок, класифікованих як шкідливі поведінки (атаки);
2. TN (true negatives) – кількість нешкідливих поведінок, класифікованих як нешкідливі поведінки;
3. FP (false positives) – кількість шкідливих поведінок (атак), класифікованих як нешкідливі поведінки (помилки першого роду, хибні спрацювання);
4. FN (false negatives) – кількість класифікованих атак як нешкідливі поведінки (невиявлення, помилки другого роду).

Для оцінки даних і висновків використовуються чотири метрики.

Чутливість – це відсоток передбачених шкідливих програм, які фактично є шкідливою програмою, яка правильно виявлена:

$$\text{Чутливість} = \frac{TP}{TP+FP}. \quad (6)$$

Специфічність – це відсоток реальних шкідливих елементів, визначених у моделі. Де TP являє собою кількість деструктивних файлів, які належним чином класифіковані, FP – це ряд доброякісних файлів, які помилково класифіковані як шкідливі програми, і FN – це кількість шкідливих файлів, які помилково класифіковані як доброякісні файли. Для виявлення шкідливих програм з великого набору даних, потрібно максимізувати чутливість і специфічність ідентифікації шкідливого програмного забезпечення:

$$\text{Специфічність} = \frac{TP}{TP+FN}, \quad (7)$$

$$\text{Достовірність} = \frac{TP+TN}{TP+TN+FP+FN}. \quad (8)$$

F-міра вважається зібраною основою реалізації для оцінки, яка є комбінацією обох критеріїв. F-міру можна розглядати як зважене середнє значення точності інтерпретації, відгуку, і, як наслідок, високої точності виклику в F-мірі. Достовірність також використовується як критерій вимірювання для правильної ідентифікації зразка зловмисного програмного забезпечення.

$$\text{F-міра} = \frac{2*\text{Відгук}*\text{Точність}}{\text{Відгук}+\text{Точність}}. \quad (9)$$

Результати, отримані від запропонованого методу, представлено в таблиці 5.

Таблиця 5

Результати роботи запропонованого методу

Чутливість, %	Специфічність, %	Достовірність, %	F-міра, %
98,43	96,39	97,55	97,39

Таким чином, запропонований метод продемонстрував можливість ідентифікації шпигунського програмного забезпечення з достовірністю до 97,55 %.

Висновки. В роботі представлено подальший розвитку методу ідентифікації шпигунського програмного забезпечення в комп'ютерних системах, який дозволяє виявляти усі типи, і відрізняється від відомих тим, що забезпечує принцип проактивності та базується на механізмах машинного навчання з підкіпленням. Запропонований метод ідентифікації шпигунського програмного забезпечення використовує аналіз поведінки програмного забезпечення в комп'ютерних системах.

Метод ґрунтується також на аналізі даних із залучення апарату машинного навчання, зокрема навчання з підкріпленням, який оперує з поняттями значень стабільності для виявлення ШПЗ. Запропонований метод, використовує рівняння отримання нової інформації, здійснює обчислення величини впливу кожної з властивостей програмного забезпечення в потрібній категорії, а завдяки ефективності стабільності ЕАХ, значення присвоюється до нього в якості нагороди.

Експериментальні дослідження продемонстрували здатність запропонованого методу до ідентифікації шпигунського програмного забезпечення з високою достовірністю (до 97,55%).

Література

1. Check Point Research. The 2020 Cyber Security Report. Available at: https://research.checkpoint.com/2020/the-2020-cyber-security-report/ (accessed 25.03.2020).

2. Kharchenko, V., Illiashenko, O., Brezhnev, E., Boyarchuk, A., Golovanevskiy, V.A. (2014). Security Informed Safety Assessment of Industrial FPGA-Based Systems.

3. FBI. Cyber Crime. Available at: https://www.fbi.gov/investigate/cyber (accessed 25.03.2020).

4. Лисенко С.М. Методи виявлення бот-мереж в комп'ютерних системах / С.М. Лисенко, К.Ю. Бобровнікова, В.С. Харченко // Сучасні інформаційні системи. – 2019. – Т. 3. № 4. – С. 87–95.

5. Rai A. D., Ward S., Roy S. Warnick, Vulnerable links and secure architectures in the stabilization of networks of controlled dynamical systems. In: Proceedings of the 2012 American Control Conference, Montreal, QC (Canada, 2012). 2012. P. 27–29.

6. Naval S., Laxmi V., Rajarajan M., et al. Employing program semantics for malware detection. IEEE Trans Inform Forens Secur. 2015. Vol.10. No.12. P. 2591–2604. https://doi.org/10.1109/TIFS.2015.2469253

7. Wang S.W., Wang B.S., Yong T., et al. Malware clustering based on SNN density using system calls. Proc 1st Int Conf on Cloud Computing and Security, 2015. P. 181–191. https://doi.org/10.1007/978-3-319-27051-7_16

8. Wang Z., Fei M., Du D., Zheng M. Decentralized event-triggered average consensus for multi-agent systems in CPSs with communication constraints, IEEE/CAA J. Autom. Sin. 2015. Vol. 2. No. 3. P. 248–257.

9. Liu L., Wang B.S., Yu B., et al. A novel selective ensemble learning based on K-means and negative correlation. Proc 2nd Int Conf on Cloud Computing and Security, 2016. P. 578–588. https://doi.org/10.1007/978-3-319-48674-1_51.

10. Liu L., Esmalifalak M., Ding Q., Emesih V.A., Han Z. Detecting false data injection attacks on power grid by sparse optimization, IEEE Trans. Smart Grid. 2014. Vol. 5. No. 2. P. 612–621.

11. Liu Y., Xin H., Qu Z., Gan D. An attack-resilient cooperative control strategy of multiple distributed generators in distribution networks, IEEE Trans. Smart Grid. 2016. Vol. 7. No. 6. P. 2923–2932.

12. Zhao Z.Q., Wang J.F., Bai J.R. Malware detection method based on the control-flow construct feature of software. IET Inform Secur. 2014. Vol. 8. No. 1. P. 18–24. https://doi.org/10.1049/iet-ifs.2012.0289.

13. Ding D.G., Wei S., Zhang Y., Liu F.E. Alsaadi, On scheduling of deception attacks for discrete-time networked systems equipped with attack detectors, Neurocomputing. 2017. Vol. 219. P. 99–106.

14. Ding D., Shen Y., Song Y., Wang Y., Gen J. Recursive state estimation for discrete time-varying stochastic nonlinear systems with randomly occurring deception attacks, Int. Syst. 2016. Vol. 45. No. 5. P. 548–560.

15. Ding D., Wang Z., Ho D.W.C., Wei G., Observer-based event-triggering con- sensus control for multi-agent systems with lossy sensors and cyber attacks, IEEE Trans. Cybern. 2017. Vol. 47. No. 8. P. 1936–1947.

16. Cen L., Gates C. S., Si L., et al., A probabilistic discriminative model for Android malware detection with decompiled source code. IEEE Trans Depend Sec Comput. 2015. Vol.12. No.4. P. 400–412. https://doi.org/10.1109/TDSC.2014.2355839.

17. Zhang H., Yao D.F., Ramakrishnan N., et al. Causality reasoning about network events for detecting stealthy malware activities. Comput Secur. 2016. Vol. 58. P. 180–198. https://doi.org/10.1016/j.cose.2016.01.002

18. Zhang H., Cheng P., Shi L., Chen J. Optimal denial-of-service attack scheduling in cyber-physical systems, Technical Report, Zhejiang University, 2015. (On-line). http://www.sensornet.cn/heng/Hengestimation Full.pdf.

19. Zhang H., Shu Y., Cheng P., Chen J. Privacy and performance trade-off in cyber-physical systems, IEEE Netw. 2016. Vol. 30. No. 2. P. 62–66.

20. Maruthupandi, J., Vimala Devi, K. Multi-label text classification using optimized feature sets. Int. J. of Data Mining, Modelling and Management, 2017. Vol. 9, No. 3, . 237–248.

21. Buffet, O., Dutech, A., Charpillet, F. Shaping multi-agent systems with gradient reinforcement learning, Autonomous Agents and Multi-Agent Systems, 2007, Vol. 15, No. 2, p. 197–220.

22. Azhagusundari, B., Thanamani, A.S. Feature selection based on information gain, International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2013,Vol. 2, No. 2, p. 2278–3075.

23. Morales, E.F. and Zaragoza, J.H. An introduction to reinforcement learning', in Sucar, L.E., Morales, E.F. and Hoey, J. (Eds.): Chapter in Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions, 2012, Vol. 19, No. 4, p. 639–668, IGI Global.

24. Canadian Institute for Cybersecurity. Malware dataset, https://www.unb.ca/cic/datasets/botnet.html (April 7, 2020).

Надійшла / Paper received: 04.05.2020 Надрукована / Paper Printed : 01.06.2020