КОМП'ЮТЕРНІ НАУКИ, ІНФОРМАЦІЙНІ ТЕХНОЛОГІ,
СИСТЕМНИЙ АНАЛІЗ ТА КІБЕРБЕЗПЕКА

DOI 10.31891/2307-5732-2020-287-4-7-11
УДК 004.93

К.Ю. БОБРОВНІКОВА, Д.О. ДЕНИСЮК
Хмельницький національний університет

МЕТОД ВИЯВЛЕННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ШЛЯХОМ АНАЛІЗУ МЕРЕЖНОГО ТРАФІКУ ТА ПОВЕДІНКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В КОМП'ЮТЕРНИХ СИСТЕМАХ

В роботі представлено метод виявлення шкідливого програмного забезпечення шляхом аналізу мережного трафіку та поведінки програмного забезпечення в комп'ютерних системах. Метод ґрунтується на класифікації множин API-викликів, вилучених з побудованих графів потоків керування для програмних додатків, та використовує аналіз DNS-трафіку комп'ютерної мережі. В якості класифікатора застосована комбінація глибокої нейронної та рекурентної нейронної мереж. Застосування розробленого методу дозволило підвищити достовірність виявлення шкідливого програмного забезпечення в комп'ютерних системах.

Ключові слова: шкідливе програмне забезпечення, комп'ютерні системи, достовірність виявлення, кібератака, мережний трафік.

K. BOBROVNIKOVA, D. DENYSIUK
Khmelnytskyi National University

METHOD FOR MALWARE DETECTION BASED ON THE NETWORK TRAFFIC ANALYSIS AND SOFTWARE BEHAVIOR IN COMPUTER SYSTEMS

The paper presents a method for malware detection by analyzing network traffic and software behavior in computer systems. The method is based on the classification of API call sets extracted from the constructed control flow graphs for software applications, and based on the analysis of DNS traffic of the computer network. As a classifier a combination of deep neural network and recurrent neural network is used. The proposed method consists of two stages: the deep neural network and the recurrent neural network learning stage and the malware detecting stage. The steps of the malware detecting are: construction of a set of graphs of control flows for software applications in computer system; construction of the set of used APIs based on the set of graphs of control flows; construction of a set of frequencies of API on the basis of a set of graphs of control flows; construction of a set of API sequences based on a set of graphs of control flows; extraction of features from network DNS-traffic; construction of a test sample; processing a test sample using a deep neural network; processing a test sample using a recurrent neural network; combinations of malware detection results using a deep neural network and a recurrent neural network; malicious software removal. Experimental studies were carried out, the results of which showed that the use of a deep neural network makes it possible to obtain the reliability of malicious software detection at the level from 94.75 to 98.66%, the use of a recurrent neural network - from 96.63% to 99.17%. The combination of the results of the classification of deep and recurrent neural networks allows achieving the best results, in which the reliability of malicious software detection is at the level of 97.29 to 99.42%. The usage of the developed method allowed to increase the reliability of malware detection in computer systems.

Keywords: malware, computer systems, detection efficiency, cyberattack, network traffic.

Вступ

Останнім часом шкідливе програмне забезпечення (ШПЗ), таке як бекдори, бот-мережі, вимагачі, шпигунське програмне забезпечення, дроппери, криптомайнери, трояни, рекламне програмне забезпечення стають все більш актуальною загрозою. Протягом минулих років третина всіх атак – від створення бот-мереж до викрадення банківських даних, вчинення клік-шахрайств чи загрози репутації – здійснювалась прихованими програмами. При цьому застосовуються як нові методи інфікування комп'ютерних систем (КС), так і перевірені схеми розповсюдження [1–3]. Тому метою роботи є підвищення достовірності виявлення ШПЗ в комп'ютерних системах шляхом розроблення методу їх виявлення.

Пов'язані роботи

Сьогодні в наукових джерелах широко представлені різноманітні підходи до виявлення ШПЗ. Зокрема, в [4] запропоновано підхід для захисту пристроїв IoT від атак з локальних комп'ютерних систем. Підхід заснований на аналізі поведінки ШПЗ та застосовує методи глибокого навчання з використанням хмарних технологій. Для одержання інформації про поведінку шкідливих програм на основі аналізу API викликів здійснюється побудова графів поведінки. Для вилучення ознак з графів поведінки використовується нейронна мережа архітектури автокодувальник. В [5] запропоновано підхід виявлення ШПЗ, заснований на аналізі дозволів та намірів. З метою виявлення ШПЗ використовується декілька пов'язаних класифікаторів: таблиці рішень, багатошаровий персептрон та дерева рішень для визначення таких оцінок, як середнє значення ймовірностей, добуток ймовірностей та більшість голосів. У роботі [6] представлено метод виявлення ШПЗ, який ґрунтується на аналізі журналів системних викликів. Результати експериментів із застосуванням методу показали високу точність виявлення. Проте важливим недоліком методу є те, що він не враховує здатність деяких шкідливих програмних додатків ідентифікувати середовище типу пісочниці. В [7] запропонований підхід виявлення ШПЗ, що використовує глибоку

згорткову нейронну мережу. Класифікація ШПЗ здійснюється на основі статичного аналізу необроблених послідовностей коду з дизасембльованих програм.

Запропонований в [8] метод для виявлення ШПЗ використовує набір ознак, таких як апаратне забезпечення, дозволи, компоненти програмного додатку, відфільтровані наміри, опкоди та рядки, що вилучаються із досліджуваних зразків. Ефективність підходу досліджено за допомогою ряду класифікаторів: ліс, що обертається, випадковий ліс, метод опорних векторів. В [9] розроблена система виявлення ШПЗ на основі статичного аналізу. Система функціонує в чотири етапи: (1) побудова графів API-викликів для кожної програми; (2) одержання послідовностей API-викликів, використовуючи унікальні вузли, та віднесення кожного виклику до певного класу, пакету чи сімейства; (3) моделювання поведінки програмного додатку шляхом побудови ланцюгів Маркова з послідовностей API-викликів; (4) класифікація додатка як нешкідливе або шкідливе програмне забезпечення на основі аналізу ймовірностей переходів, використовуваних в якості векторів ознак. В [10] розроблено систему, яка здійснює класифікацію програмних додатків на основі тріажу з урахуванням їх потенційного ризику. Для класифікації використано ймовірнісну модель, яка надає можливість прогнозування існування інформаційних потоків, та показник, що визначає значущість цих інформаційних потоків. Результати експериментів показали, що система здатна досить точно передбачити наявність інформаційних потоків шкідливих додатків і дозволяє досягти економії обчислювальних ресурсів.

Огляд літератури показав, що проблема виявлення ШПЗ є надзвичайно актуальною. Відомі методи виявлення ШПЗ демонструють високий рівень ефективності, але також мають високий показник хибних спрацювань. Загальним недоліком відомих підходів є потреба у великих обсягах обчислювальних ресурсів та те, що вони не здатні адаптивно реагувати на відомі та невідомі атаки, здійснені ШПЗ на комп'ютерні системи. Також розглянуті підходи мають деякі загальні недоліки, які полягають в ігноруванні упакованого ШПЗ та неможливості захистити пристрій від загроз нульового дня і шкідливих програм, здатних модифікувати себе.

Метод виявлення шкідливого програмного забезпечення шляхом аналізу мережного трафіку та поведінки програмного забезпечення в комп'ютерних системах

Розглянемо основи методу виявлення шкідливого програмного забезпечення шляхом аналізу мережного трафіку та поведінки програмного забезпечення в комп'ютерних системах.

Запропонований метод складається з двох етапів: етап навчання глибокої нейронної мережі та рекурентної нейронної мережі та етап виявлення ШПЗ. Сформулюємо основні кроки етапу навчання:

1) формування навчальної вибірки зі зразків шкідливого і нешкідливого ПЗ;

2) побудова множини графів потоків керування для ПЗ з навчальної вибірки;

3) побудова множин використовуваних API-викликів на основі множини графів потоків керування;

4) побудова множин частот здійснення API-викликів на основі множини графів потоків керування;

5) побудова множин послідовностей здійснення API-викликів на основі множини графів потоків керування;

6) вилучення ознак зі зразків мережного DNS-трафіку;

7) побудова навчальної вибірки для навчання глибокої нейронної та рекурентної нейронної мереж;

8) навчання глибокої нейронної та рекурентної нейронної мереж.

Сформулюємо кроки етапу виявлення ШПЗ:

1) побудова множини графів потоків керування для ПЗ в КС;

2) побудова множин використовуваних API-викликів на основі множини графів потоків керування;

3) побудова множин частот здійснення API-викликів на основі множини графів потоків керування;

4) побудова множин послідовностей здійснення API-викликів на основі множини графів потоків керування;

5) вилучення ознак з мережного DNS-трафіку;

6) побудова тестової вибірки;

7) оброблення тестової вибірки за допомогою глибокої нейронної мережі;

8) оброблення тестової вибірки за допомогою рекурентної нейронної мережі;

9) комбінування результатів виявлення шкідливого ПЗ за допомогою глибокої нейронної мережі та рекурентної нейронної мережі;

10) видалення шкідливого ПЗ.

З метою аналізу поведінки програмного забезпечення в КС в запропонованому методі використовується побудова та аналіз графів потоків керування програмних додатків. Граф потоку керування представляється у вигляді орієнтованого графу, який містить інформацію про складові частини програми та переходи між ними, при цьому враховуються безумовні переходи, розгалуження, цикли, виклики функцій та виключення. Граф потоку керування описує потоки керування, які можуть виникнути при виконанні програми, при цьому кожен API-виклик є вузлом графу. Отже, для кожного екземпляру програмного забезпечення будується граф потоку керування, і з кожного графу потоку керування вилучаються наступні множини API-викликів:

1) множина використовуваних API-викликів (які саме API-виклики здійснює додаток), $S_u = \{API_i\}_{i=1}^{N}$, де N – кількість різних API-викликів;

2) множина частот здійснення API-викликів (скільки разів здійснюється кожен API-виклик),

$S_f = \{API_i, n_i\}_{i=1}^{N}$, де n_i – кількість разів здійснення API-виклику;

3) множина послідовностей здійснення API-викликів (порядок здійснення API-викликів в часі), $S_s = \{API_k\}_{k=1}^{M}$, де M – загальна кількість всіх здійснених API-викликів.

Для побудови множин API-викликів на основі графів потоків керування використовується пошук в глибину (DFS).

З метою виявлення ШПЗ також залучається множина вилучених з DNS-трафіка ознак, що можуть свідчити про активність шкідливого програмного забезпечення в мережі (табл. 1).

Таблиця 1

Множина ознак DNS-трафіка та діапазони значень, що можуть свідчити про активність ШПЗ в мережі

№ зп	Назва ознаки	Значення ознаки	Діапазон значень
1	l_N	довжина запитуваного доменного імені d	$l_N \in [75, 255]$
2	n_U	кількість унікальних символів в d	$n_U \in (27, 37]$
3	e_N	ентропія d	$e_N \geq f_{Ebn}$, f_{Ebn} – функція залежності ентропії від довжини поля, n – основа кодування
4	f_{UR}	використання рідко вживаних типів DNS-записів, які зазвичай не використовується типовим клієнтом (TXT, KEY, NULL тощо)	1
5	n_{IP}	кількість IP-адрес, пов'язаних з d	$n_{IP} \in (5, \infty)$
6	s_{IP}	середня дистанція між IP-адресами, пов'язаними з d	$s_{IP} \in (65535, \infty)$
7	n_A	кількість A-записів, що відповідають d, у вхідному DNS-повідомленні	$n_A \in (5, \infty)$
8	s_A	середня дистанція між IP-адресами в множині A-записів для d у вхідному DNS-повідомленні	$s_A \in (65535, \infty)$
9	n_{UA}	кількість унікальних IP-адрес в множинах A-записів, що відповідають d, у вхідних DNS-повідомленнях	$n_{UA} \in (8, \infty)$
10	s_{UA}	середня дистанція між унікальними IP-адресами в множинах A-записів, що відповідають d, у DNS-повідомленнях	$s_{UA} \in (65535, \infty)$
11	n_D	кількість доменних імен, які спільно використовують IP-адресу, що відповідає d	$n_D \in [8, \infty]$
12	$t_{mod}, t_{med}, t_{aver}$	TTL-період, мода, медіана, середнє арифметичне значення	$t_{mod}, t_{med}, t_{aver} \in [0, 900]$
13	f_S	ознака успішності DNS-запиту	0

З перерахованого набору ознак формуються вектори ознак, які є вхідними даними для класифікаторів. Результатом класифікації є приналежність ПЗ до одного з двох класів – шкідливе (1) або нешкідливе (0) програмне забезпечення.

Експериментальні дослідження

З метою визначення достовірності виявлення ШПЗ запропонованим методом було проведено ряд експериментів. Для формування навчальної та тестової вибірок було зібрано 12208 шкідливих додатків та зразків їх трафіку з відкритих джерел [11], також було використано 11961 зразок нешкідливого програмного забезпечення. Для тестування було використано 6203 шкідливих додатки та зразки їх трафіку та 5962 зразки нешкідливого ПЗ, а решта зразків були використані для навчання класифікаторів. В першому експерименті в якості класифікатора було використано глибоку нейронну мережу (Deep neural network, DNN), в другому – рекурентну нейронну мережу архітектури Long short-term memory, LSTM, а в третьому було скомбіновано результати класифікації обох нейронних мереж за допомогою бустингу. В якості вагових коефіцієнтів застосовано достовірності виявлення кожним класифікатором, а для обчислення остаточного результату – метод зваженої суми. Якщо результат класифікації перевищував порогове значення, то програмне забезпечення було класифіковане як шкідливе, а інакше – як нешкідливе ПЗ. В якості порогового значення встановлено величину, що дорівнює 0,5.

Результати проведених експериментів із застосуванням глибокої нейронної мережі, рекурентної нейронної мережі та комбінації результатів класифікації глибокої та рекурентної нейронних мереж подані в табл. 2 та на рис. 1, де *TP* (true positive) – істинно-позитивний результат класифікації, вірно класифіковані шкідливі зразки; *TN* (true negative) – істинно-негативний результат, вірно класифіковані нешкідливі зразки; *FN* (false negative) – хибно-негативний результат, шкідливі зразки, помилково класифіковані як нешкідливі; *FP* (false positive) – хибно-позитивний результат, нешкідливі зразки, помилково класифіковані як шкідливі; *ACC* (accuracy) – достовірність виявлення ШПЗ, $ACC = (TP + TN)/(TP + TN + FP + FN)$.

Таблиця 2

Результати виявлення ШПЗ із застосуванням глибокої, рекурентної нейронних мереж та комбінації результатів класифікації глибокої та рекурентної нейронних мереж

Назва ШПЗ	DNN					LSTM					Комбінація результатів класифікації DNN та LSTM				
	TP	TN	FN	FP	ACC	TP	TN	FN	FP	ACC	TP	TN	FN	FP	ACC
Dridex	592	593	25	1	97,85%	592	587	25	7	97,36%	602	585	15	9	98,02%
Emotet	586	587	62	3	94,75%	617	586	31	4	97,17%	619	586	29	4	97,33%
Gh0st	567	570	31	7	96,77%	568	577	30	0	97,45%	579	576	19	1	98,30%
IcedID	590	594	29	4	97,29%	590	586	29	12	96,63%	599	596	20	2	98,19%
Kovter	590	592	14	2	98,66%	595	593	9	1	99,17%	597	594	7	0	99,42%
NanoCore	594	590	16	15	97,45%	599	600	11	5	98,68%	599	603	11	2	98,93%
Pushdo	595	602	49	12	95,15%	612	614	32	0	97,46%	618	614	26	0	97,93%
Trickbot	592	594	36	1	96,97%	589	593	39	2	96,65%	605	595	23	0	98,12%
WannaCry	588	588	27	15	96,55%	594	589	21	14	97,13%	590	595	25	8	97,29%
ZeuS	591	592	29	0	97,61%	584	591	36	1	96,95%	599	592	21	0	98,27%

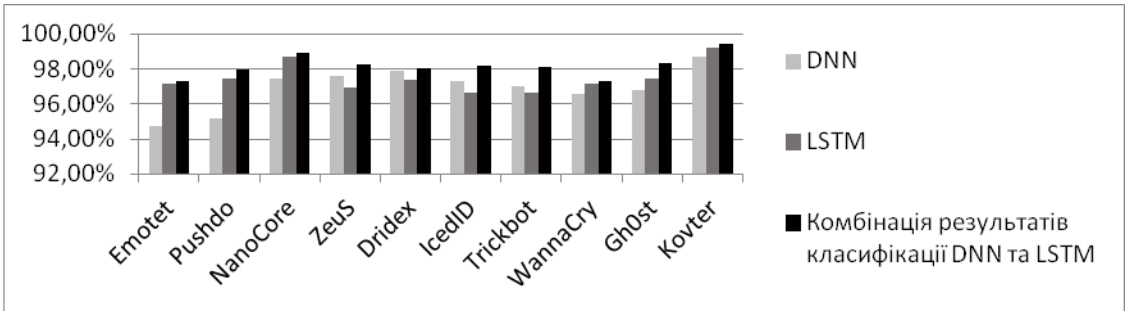

Рис. 1. Результати виявлення ШПЗ із застосуванням глибокої нейронної мережі, рекурентної нейронної мережі та комбінації результатів класифікації глибокої та рекурентної нейронних мереж

Результати проведених експериментів показали, що при застосуванні глибокої нейронної мережі було одержано достовірність виявлення ШПЗ на рівні від 94,75 до 98,66%, при застосуванні рекурентної нейронної мережі – від 96,63% до 99,17%. При комбінації результатів класифікації глибокої та рекурентної нейронних мереж було одержано найкращий результат, за якого достовірність виявлення шкідливого програмного забезпечення є на рівні від 97,29 до 99,42%.

Висновки

В роботі представлено метод виявлення ШПЗ шляхом аналізу мережного трафіку та поведінки програмного забезпечення в КС, який ґрунтується на класифікації множин використовуваних API-викликів, частот здійснення API-викликів і послідовностей здійснення API-викликів, вилучених з побудованих графів потоків керування для програмних додатків, а також ознак, вилучених з DNS-трафіку мережі. Наведено результати проведених експериментів, які показали, що при застосуванні комбінації результатів класифікації за допомогою глибокої та рекурентної нейронних мереж достовірність виявлення шкідливого програмного забезпечення є на рівні від 97,29 до 99,42%. Отже, застосування розробленого методу дозволить підвищити достовірність виявлення ШПЗ в КС.

Література

1. McAfee Labs Threats Reports. Insights into malware, ransomware, and other cybersecurity threats from the McAfee threat research team. URL: https://www.mcafee.com/enterprise/ru-ru/threat-center/mcafee-labs/reports.html. – 2.07.2020.

2. 2020 State of Malware Report. URL: https://resources.malwarebytes.com/files/2020/0 2/2020_State-of-Malware-Report.pdf. – 2.07.2020.

3. Securelist. Statistics. URL: https://statistics.securelist.com/en. – 2.07.2020.

4. Xiao F. Malware detection based on deep learning of behavior graphs / F. Xiao, Z. Lin, Y. Sun, Y. Ma // Mathematical Problems in Engineering. – 2019.

5. Idrees F. Pindroid: a novel android malware detection system using ensemble learning methods / F. Idrees, M. Rajarajan, M. Conti, T. Chen, Y. Rahulamathavan // Computers & Security. – 2017. – Vol. 68. – P. 36–46.

6. Chaba S. Malware Detection Approach for Android systems Using System Call Logs / S. Chaba, R. Kumar, R. Pant, M. Dave // arXiv preprint arXiv:1709.08805. – 2017.

7. McLaughlin N. Deep android malware detection / N. McLaughlin, J. Martinez del Rincon, B. Kang // Proc. of the Seventh ACM on Conference on Data and Application Security and Privacy. – 2017. – P. 301–308.

8. Varsha M. Identification of malicious android app using manifest and opcode features / M. Varsha, P.

Vinod, K. Dhanya // Journal of Computer Virology and Hacking Techniques. – 2016. – Vol. 13, Issue 2. – P. 125–138.

9. Onwuzurike L. MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models / L. Onwuzurike, E. Mariconti, P. Andriotis, E. D. Cristofaro // ACM Trans. Priv. Sec. – 2019. – Vol. 22, No. 2. – P. 1–34.

10. Mirzaei O. Triflow: Triaging android applications using speculative information flows / O. Mirzaei, G. Suarez-Tangil, J. Tapiador, J.M. de Fuentes // Proc. of the 2017 ACM on Asia Conference on Computer and Communications Security. – 2017. – P. 640–651.

11. Canadian Institute for Cybersecurity. Botnet dataset. URL: https://www.unb.ca/cic/datasets/botnet.html – 5.12.2019.

Рецензія/Peer review : 15.10.2020 р. Надрукована/Printed : 02.11.2020 р.