РОЗДІЛ ІІ. ІНФОРМАЦІЙНО-КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 004.056.57
DOI: 10.25140/2411-5363-2019-4(18)-85-91

Володимир Казимир, Ігор Карпачев, Владислав Сіпаков

ДИНАМІЧНИЙ АНАЛІЗ ПОСЛІДОВНОСТЕЙ API-ВИКЛИКІВ ОС ANDROID

Актуальність теми дослідження. *На сьогодні набуло значного поширення використання користувачами мобільних пристроїв та додатків з метою зберігання приватної та конфіденційної інформації. Поряд з цим, існують методи та шляхи поширення шкідливого програмного забезпечення (ПЗ) в операційній системі (ОС) Android. Для ефективної боротьби з ними постає необхідність розробки нових покращених підходів до виявлення шкідливого ПЗ в ОС Android. У статті розглянуто новий метод динамічного аналізу додатків, який дозволить покращити існуючу систему безпеки.*

Постановка проблеми. *У галузі забезпечення ефективної та безпечної роботи мобільних пристроїв функціональна та інформаційна безпека розглядаються як дві фундаментальні складові, що взаємодоповнюють одна одну. Одним із найефективніших способів отримання зловмисником доступу до конфіденційної інформації є використання ОС Android. Одним із засобів підвищення надійності роботи є розробка моделей безперервного динамічного захисту.*

Аналіз останніх досліджень і публікацій. *Розглянуто останні публікації у відкритому доступі, включаючи дані Google Malware Project, PScout Android Permissions Mappings та Axplorer Permissions Mappings та роботу Android Malware Detection Using Permission Analysis.*

Виділення недосліджених частин загальної проблеми. *Розробка та математичне обґрунтування моделей безперервного аналізу API-викликів в ОС Android.*

Постановка завдання. *Запропонувати базову модель захисту ОС Android, що ґрунтується на аналізі послідовностей API-викликів.*

Виклад основного матеріалу. *У статті наведено метод отримання та аналіз послідовностей API-викликів ОС Android за допомогою інструментарію Frida. Також наведено найбільш репрезентативні результати тестування запропонованого методу.*

Висновки відповідно до статті. *Запропоновано метод динамічного аналізу шкідливого ПЗ з використанням інструментарію для реверс-інжинірингу Android та IOS додатків – Frida. Запропонований метод динамічного аналізу API-викликів можна використовувати для покращення існуючих систем аналізу APK, а метод дослідження вірусних послідовностей можна поширити для всіх характерних вірусних груп.*

Ключові слова: *безпека; безпека Android; API-виклики; зловмисне ПЗ; динамічний аналіз; дозволи; Frida.*
Рис.: 5. Бібл.: 9.

Актуальність теми дослідження. Використання користувачами мобільних пристроїв та додатків з метою зберігання приватної та конфіденційної інформації на сьогодні набуло значного поширення. Поряд з цим існують методи та шляхи поширення шкідливого програмного забезпечення в ОС Android.

Завдяки зростаючій популярності Android за останні кілька років зловмисне програмне забезпечення, спрямоване на платформу, значно зросло. Згідно з останнім звітом McAfee [8], щороку з'являється близько 2,5 млн нових зразків шкідливих програм Android, таким чином збільшуючи загальну кількість зразків шкідливих програм до 12 млн.

Зловмисне програмне забезпечення для Android можна знайти в різних додатках, таких як ігрові програми, банківські програми, додатки соціальних медіа, освітні програми та додаткові програми. Програми, заражені зловмисним програмним забезпеченням, можуть мати доступ до конфіденційної інформації, надсилати текстові повідомлення до VIP номерів без погодження з користувачем або навіть встановлювати руткіт на пристрій, що дозволяє завантажувати та виконувати будь-який код, який розробник зловмисного програмного забезпечення хоче розгорнути, тощо.

Для ефективної боротьби зі зловмисним програмним забезпеченням постає необхідність розробки нових покращених підходів до виявлення шкідливого ПЗ в ОС Android. У статті розглянуто новий метод динамічного аналізу додатків на предмет виявлення шкідливого програмного забезпечення, який дозволить покращити існуючу систему безпеки.

Постановка проблеми. Проблеми статичного аналізу програмного забезпечення ОС Android пов'язані з недосконалістю багатьох факторів виявлення зловмисного ПЗ. Ключовою проблемою статичного аналізу дозволів основаного на стандартному методі

LSI, який було запропоновано в роботі Android Malware Detection Using Permission Analysis [6], є неможливість виявлення небезпечних частин коду apk.

Аналіз останніх джерел і публікацій. У роботі Android Malware Detection Using Permission Analysis [6], що ґрунтується на стандартному методі LSI, використовується статичний аналіз ПЗ. Замість статичного аналізу пропонується зробити динамічний аналіз послідовностей викликів. Динамічний аналіз роботи мобільних додатків обов'язково включає у себе блок забезпечення функціональної безпеки, що здійснює моніторинг потенційно небезпечних послідовностей та ідентифікує відповідні події, які можуть призвести до втрати даних, доступу до конфіденційних даних сторонніх осіб або блокування сторонніми особами належного доступу. На базовому етапі отримання послідовностей API-викликів «Android Permission» [7], є визначення API-викликів, що застосовує програму на основі мапінгів та подальше отримання послідовностей API-викликів на основі інструментарію Frida [3].

Виділення недосліджених частин загальної проблеми. Розробка та математичне обґрунтування моделей безперервного аналізу API-викликів в ОС Android.

Постановка завдання. Мета статті полягає в описі нового методу динамічного аналізу послідовностей API-викликів ОС Android.

Виклад основного матеріалу.

Метод динамічного аналізу. Станом на 2019 рік, Google Android є лідером мобільних операційних систем, займаючи майже 80 % ринку. Понад 65 млрд завантажень було зроблено з офіційного магазину ігор Google [5], а нині існує понад 1 млрд пристроїв Android у всьому світі. За даними Statista [4], до 2021 року по всьому світу буде доставлено близько 1,5 млрд пристроїв Android.

Ця стаття має продовжити дослідження, розпочате в роботі [6]. Схема статичного пошуку API-викликів наведена на рис. 1.

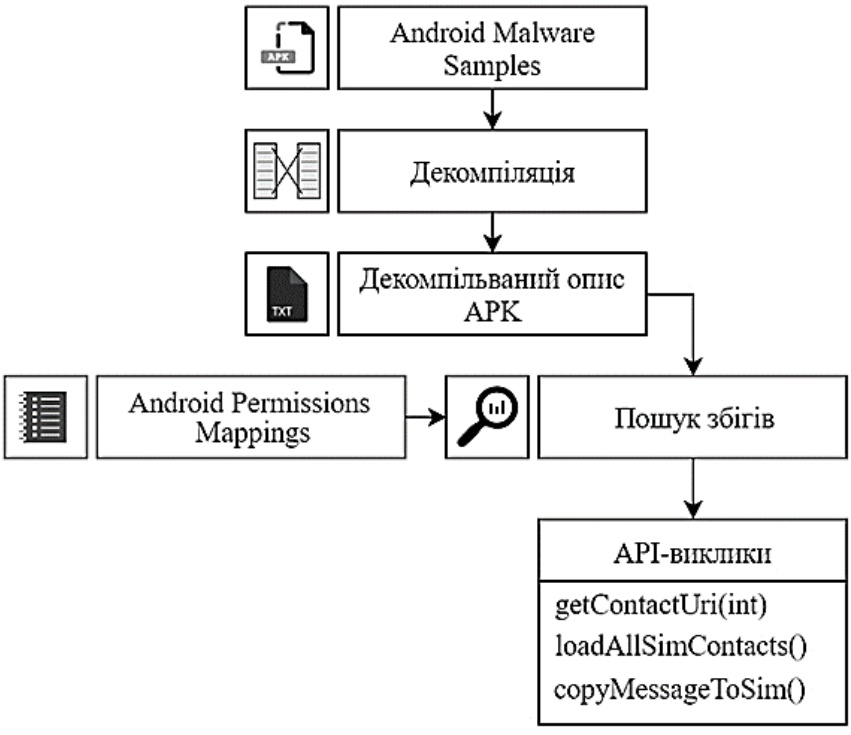

Рис. 1. Схема статичного пошуку API-викликів

Замість наявного підходу пропонується зробити динамічний аналіз послідовностей викликів, які використовують Android Permissions. Алгоритм динамічного аналізу

послідовностей API-викликів має стати підсистемою мережевого ресурсу, що розповсюджує додатки для мобільних пристроїв на основі ОС «Android».

Базова схема роботи аналізатора програмного додатка складається з таких пунктів:

1) статичний пошук API-викликів;
2) динамічне отримання послідовності за допомогою інструментарія Frida;
3) побудова графу поведінки;
4) глибинний аналіз субграфів;
5) стиснення графу;
6) отримання результатів ймовірності зловмисного ПЗ.

Для визначення API-викликів, що може застосовувати додаток, запропоновано зробити пошук на основі PScout Android Permissions Mappings та Axplorer Permissions Mappings.

Наприклад, для android.permission.READ_CONTACTS існує 57 API-викликів, серед яких найбільш поширеними є:

1) android.net.Uri addToMyContactsGroup(android.content.ContentResolver,long);
2) android.net.Uri getContactUri(int);
3) android.net.Uri getUriToQuery();
4) android.database.Cursor loadAllSimContacts(int);
5) android.widget.QuickContactBadge:
   void assignContactFromEmail(java.lang.String,boolean,android.os.Bundle).

Схема запропонованого методу динамічного отримання послідовностей API-викликів наведена на рис. 2.

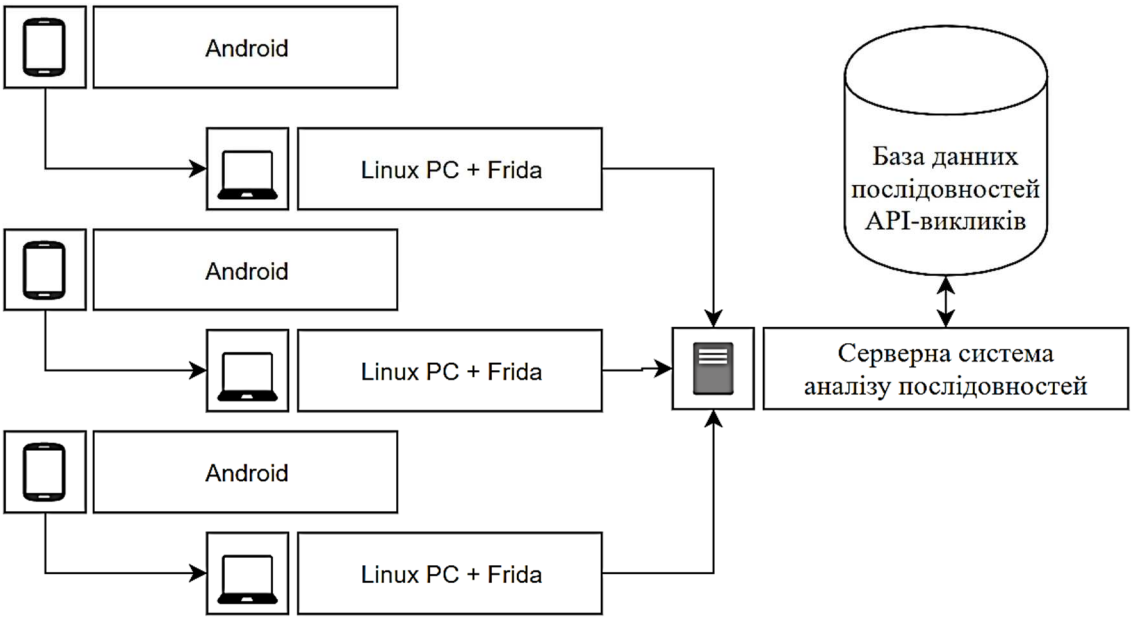

Рис. 2. Схема динамічного отримання послідовностей API-викликів

Для отримання послідовності API-викликів створено динамічний аналізатор на основі інструментарію Frida - Dynamic Instrumentation Toolkit for Developers, Reverse-engineers, and Security researchers, який дозволяє перехоплювати API-виклики та робити ін'єкції власного коду в працюючу APK. Отримана за допомогою динамічного аналізатора послідовність API-викликів, що включає передачу конфіденційних даних, наведена на рис. 3.

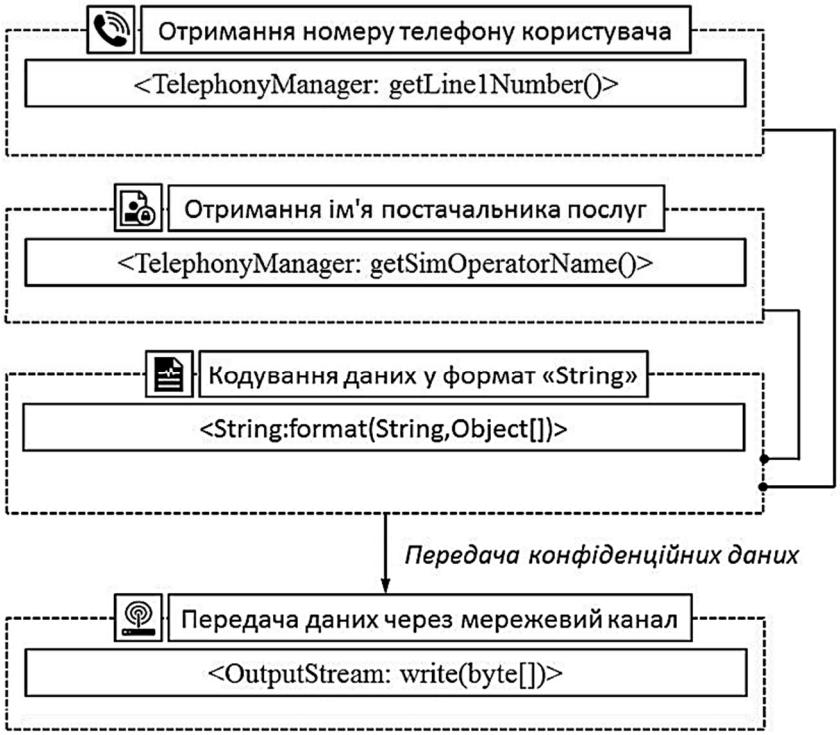

Рис. 3. Послідовності API-викликів, що включає передачу конфіденційних даних

Програмний код однієї з отриманих послідовностей включає у себе такі кроки, що базуються на основі типових API-викликів:

– отримання номера телефона користувача «getLine1Number ()»;
– отримання ім'я постачальника послуг «getSimOperatorName ()»;
– кодування даних у формат «String» (String, byte [])];
– передача даних через мережевий канал«network (write(byte[])».

Фіксуючи API-виклики з вірусних APK проєкту Android Malware Genome project [7], була сформована база данних вірусних послідовностей API-викликів. Наведені характерні для permissions API-виклики взято з досліджень Android Permissions Mappings [9] та [2].

Дослідження вірусних послідовностей. Запропонована математична модель визначення ймовірності вірусної послідовності API-викликів базується на розділенні досліджуваного інтервалу часу на i інтервалів (де $i \in [1; I]$), при цьому час, за який отримується послідовність визначається через змінну $t_i \in [t_1; t_I]$. У такому разі ймовірність виникнення вірусної послідовності з небезпечних додатків може бути визначена через рівняння:

$$P(t_i) = \frac{n_i}{N}, \tag{1}$$

де n_i – кількість вірусних послідовностей API-викликів, а N – повна кількість зафіксованих послідовностей API-викликів. Ці послідовності отримані з 246 APK проєкту Android Malware Genome Project [7]

$$P(t_i) = \frac{21880}{32657} = 0,67, \tag{2}$$

Отримання вірусної послідовності в різні інтервали часу t – незалежні події, ймовірність отримання за кожен квант $p = 0,67$.

Розрахуємо залежність знаходження ймовірності вірусної послідовності від її довжини. Для розподілу ймовірності отримання вірусної послідовності до першого точного збігу будемо використовувати геометричний закон розподілу.

$$P(X=k)=pq^{k-1},\ k=1,2,3,K, \tag{3}$$

де $p=P(A)$ – ймовірність появи події A (вірусної послідовності API-викликів) у кожному випробуванні, $q=1-p$, $X=k$ – кількість випробувань до появи події A в серії незалежних повторних випробувань.

Графік ймовірності отримання вірусної послідовності API-викликів залежно від її довжини наведений на рис. 4.

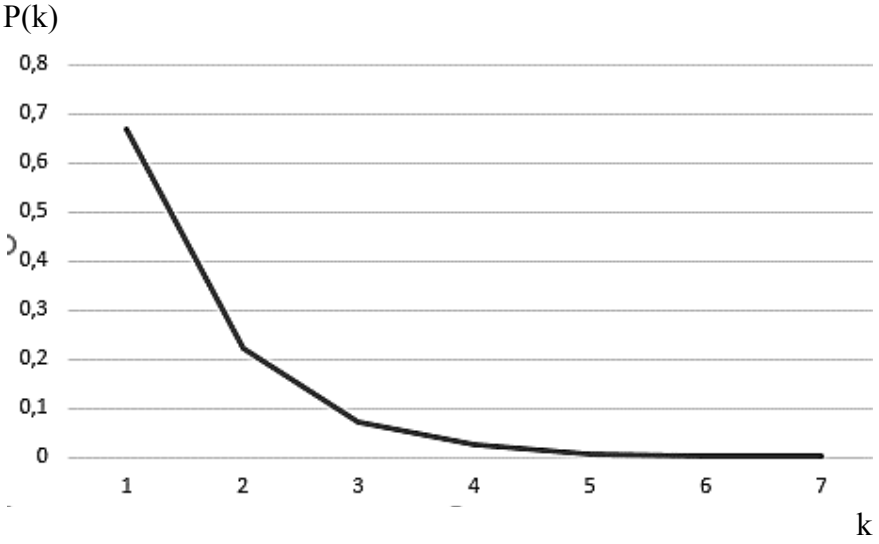

Рис. 4. Ймовірність отримання вірусної послідовності API-викликів залежно від її довжини

Нехай випадкова величина X – довжина самої послідовності. Визначимо числові характеристики даного розподілу: математичне сподівання $M(X)$, дисперсії $D(X)$ та ереднього квадратичного відхилення $\sigma(X)$:

$$M(X)=\frac{1}{p}=\frac{1}{0,67}=1,4925 \tag{4}$$

$$D(X)=\frac{q}{p^2}=\frac{0,33}{0,67^2}=0,735 \tag{5}$$

$$\sigma(x)=\frac{\sqrt{q}}{p}=\frac{\sqrt{0,33}}{0,67}=0,8573 \tag{6}$$

Для прикладу можна знайти ймовірність отримання певних API викликів троян характерних вірусів для таких послідовностей:

1) com.android.providers.contacts.ContactsProvider2;
2) android.telephony.TelephonyManager.listen;
3) getAllCellInfo;
4) getCellLocation;
5) sendDataMessage.

Результати експериментів із зафіксованою статистикою частоти викликів однієї з послідовностей, характерної для троян-вірусів, розрахованою за формулою (1), наведено на рис. 5.

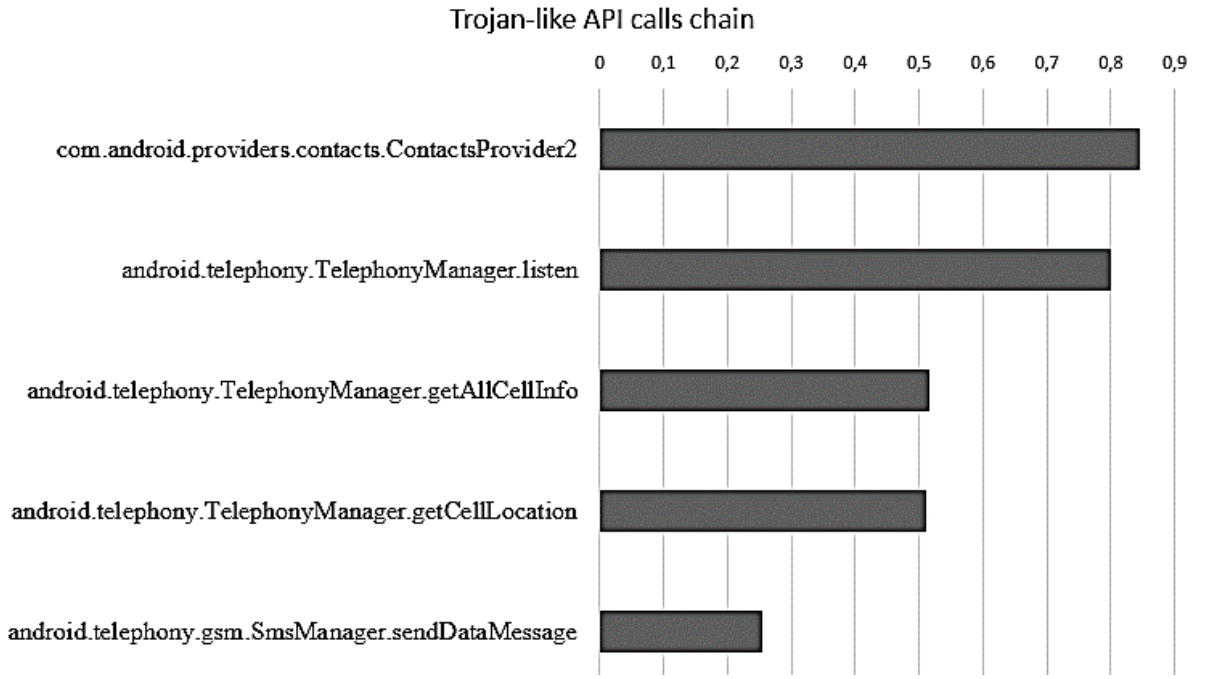

Рис. 5. Статистика частоти викликів однієї з послідовностей, характерної для троян-вірусів

Висновки відповідно до статті. Дана стаття є продовженням статті дослідження безпеки ОС Android [6]. Запропоновано метод динамічного аналізу шкідливого ПЗ з використанням інструментарію для реверс-інжинірингу Android та IOS додатків – Frida [3]. Запропонований метод динамічного аналізу API-викликів можна використовувати для покращення існуючих систем аналізу APK, а метод дослідження вірусних послідовностей можна поширити для всіх характерних вірусних груп.

Список використаних джерел

1. AndroidMalwareGenome Project. URL: http://www.malgenomeproject.org.
2. Axplorer Android Permission Mappings. A static analysis tool to study Android's application framework's internals. URL: https://github.com/reddr/axplorer.
3. Frida Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers. URL: https://www.frida.re.
4. Global smartphone shipments by OS 2016-2021 statistic. URL: https://www.statista.com/statistics/309448/global-smartphone-shipments-forecast-operating-system.
5. Google Play: number of downloads statistic. URL: https://www.statista.com/statistics/281106/number-of-android-app-downloads-from-google-play.
6. Hossain Shahriar, Mahbubul Islam, Victor Clincy. Android malware detection using permission analysis. SoutheastCon 2017. Charlotte, North Carolina, USA 30 March – 2 April 2017. 2017. P. 60-66. DOI: 10.1109/SECON.2017.7925347.
7. Manifest.permission. URL: http://developer.-ndroid.com/reference-android/Manifest.permission.html.
8. McAfee Labs Threats Report August 2019. URL: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf.
9. PScout: Analyzing the Android Permission Specification. URL: https://security.csl.toronto.edu/pscout.

References

1. AndroidMalwareGenome Project. Retrieved from http://www.malgenomeproject.org.
2. Axplorer Android Permission Mappings. A static analysis tool to study Android's application framework's internals. Retrieved from https://github.com/reddr/axplorer.

3. Frida Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers. Retrieved from https://www.frida.re.

4. Global smartphone shipments by OS 2016-2021 statistic. Retrieved from https://www.statista.com/statistics/309448/global-smartphone-shipments-forecast-operating-system.

5. Google Play: number of downloads statistic. Retrieved from https://www.statista.com/statistics/281106/number-of-android-app-downloads-from-google-play.

6. Hossain Shahriar, Mahbubul Islam, Victor Clincy. Android malware detection using permission analysis. SoutheastCon 2017. Charlotte, North Carolina, USA 30 March – 2 April 2017. 2017. P. 60-66. DOI: 10.1109/SECON.2017.7925347.

7. Manifest.permission. Retrieved from http://developer.-ndroid.com/reference-/android/Manifest.permission.html.

8. McAfee Labs Threats Report August 2019. URL: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf.

9. PScout: Analyzing the Android Permission Specification. Retrieved from https://security.csl.toronto.edu/pscout.

UDC 004.056.57

Volodymyr Kazymyr, Igor Karpachev, Vladyslav Sipakov

DYNAMIC ANALYSIS OF ANDROID API CALLS

Urgency of the research. *Storing user's private and confidential information have been widely used today by users of mobile applications. Methods and ways of spreading malware in the Android operating system are growing at the same time. A new and improved approach to Android OS access model analysis is needed for effective protection. The article deals with the development of new models of dynamic analisis, which will lead to improvement in the existing security system.*

Target setting. *In the area of ensuring the safe and efficient operation of mobile applications, functional and information security are considered as two fundamental components complementing each other. One of the most effective ways for an intruder to access confidential information is bypassing the system's single-point OS check. One of the tools to improve the reliability of work is the development of models of dynamic continuous protection.*

Actual scientific researches and issues analysis. *Recent open publications were considered, including data from Google Malware Project, PScout Android Permissions Mappings, Axplorer Permissions Mappings and article Android Malware Detection Using Permission Analysis.*

Uninvestigated parts of general matters defining. *The development and mathematical justification of the models of continuous analysis of API call chains in Android OS.*

The research objective. *Suggest a basic Android OS protection model based on the analysis of API call chains.*

The statement of basic materials. *The article provides a method for receive and analyzing API calls chains of Android OS using the Frida instrumentation toolkit. There are also most representative results of proposed method are also presented.*

Conclusions. *The article proposes a dynamic malware analysis method and describes a series of steps to ensure effective Android OS security. The proposed method for investigating sensitive API call chains can be used to improve existing APK analysis systems.*

Keywords: *security; Android security; API calls; malware; dynamic analysis; permissions; Frida.*

Fig.: 6. References: 10.

Казимир Володимир Вікторович – доктор технічних наук, професор, Чернігівський національний технологічний університет (вул. Шевченка, 95, м. Чернігів, 14035, Україна).
Kazymyr Volodymyr – Doctor of Technical Sciences, Professor, Chernihiv National University of Technology (95 Shevchenka Str., 14035 Chernihiv, Ukraine).
E-mail: vvkazymyr@gmail.com
ORCID: http://orcid.org/0000-0001-8163-1119
Scopus Author ID: 56644727300

Карпачев Ігор Ігорович – аспірант кафедри інформаційних та комп'ютерних систем, Чернігівський національний технологічний університет (вул. Шевченка, 95, м. Чернігів, 14035, Україна).
Karpachev Igor – PhD student, Department of Informational and Computer Systems, Chernihiv National University of Technology (95 Shevchenka Str., 14035 Chernihiv, Ukraine).
E-mail: benchakalaka@gmail.com
ORCID: http://orcid.org/0000-0003-1910-3264
ResearcherID: R-3626-2016

Сіпаков Владислав Сергійович – магістр кафедри інформаційних та комп'ютерних систем, Чернігівський національний технологічний університет (вул. Шевченка, 95, м. Чернігів, 14035, Україна).
Sipakov Vladyslav – master student, Department of Informational and Computer Systems, Chernihiv National Technological University (95 Shevchenka Str., 14035 Chernihiv, Ukraine).
E-mail: vladsipakovwork@gmail.com
ORCID: http://orcid.org/0000-0003-2070-2504

Казимир В., Карпачев І., Сіпаков В. Динамічний аналіз послідовностей API-викликів ОС Android. *Технічні науки та технології*. 2019. № 4 (18). С. 85-91.