УДК 004.056.57
DOI: 10.15827/0236-235X.123.543-547
Дата подачи статьи: 29.01.18
2018. Т. 31. № 3. С. 543–547

Нейросетевой метод обнаружения вредоносных программ на платформе Android

Т.М. Татарникова [1], *д.т.н., доцент, профессор, tm-tatarn@yandex.ru*
А.М. Журавлев [1], *магистрант, asn93@mail.ru*

[1] *Санкт-Петербургский государственный университет аэрокосмического приборостроения, г. Санкт-Петербург, 190000, Россия*

Непрерывный рост числа вредоносных программ, нацеленных на операционную систему Android, делает актуальной задачу их обнаружения. Отсутствие централизованного механизма распространения приложений и недостаточная эффективность существующих решений усугубляют проблему.

Экспериментально показана недостаточная эффективность существующих механизмов обнаружения вредоносных программ операционной системы Android. Необходимость автоматизированного решения, позволяющего повысить вероятность обнаружения вредоносных программ операционной системы Android, в том числе ранее неизвестных, модифицированных и обфусцированных версий уже известных программ, определила цель исследования.

В работе применялись методы классификации нейронных сетей, статического анализа кода и анализа поведения программы в виртуальной среде. Новизна предлагаемого решения состоит в использовании классификационных признаков, полученных как статическим анализом кода, так и анализом поведения программы на виртуальном устройстве. Предложенный подход устраняет недостатки существующих решений рассматриваемой проблемы.

Экспериментально доказана способность предложенного решения обнаруживать вредоносные программы, не использовавшиеся в обучении нейронной сети, а также обфусцированные экземпляры.

Практическая значимость предлагаемого решения заключается в его применении для построения систем обнаружения вредоносных программ операционной системы Android, которую можно использовать в магазине приложений Android.

Ключевые слова: *платформа Android, вредоносные программы, антивирусы, машинное обучение, нейронные сети, классификация, анализ программ.*

Google Android в настоящее время занимает лидирующие позиции на рынке мобильных *операционных систем* (ОС) [1]. Постоянно растет и количество вредоносных программ на ОС Android. Например, шпионская программа SMSVova, маскировавшаяся под системное обновление, была скачана свыше трех миллионов раз и оставалась незамеченной более двух лет. Мобильные вирусы способны шпионить за пользователями, использоваться для получения незаконных доходов, например, отправлять платные SMS или шифровать данные пользователя с целью вымогательства [2].

Помимо этих угроз, существуют и другие проблемы, связанные с недостаточной проверкой приложений, реализованных Google Play и другими магазинами, при скачивании которых пользователи устанавливают себе вредоносное ПО, замаскированное под игры и программы. Более того, пользователи с целью экономии часто прибегают к сторонним источникам программ для ОС Android, например торрентам, где приложения и вовсе могут не проверяться. Пользователи также подвергаются утечкам персональных данных, скрытым загрузкам на веб-сайтах и другим опасностям [3].

Меры противодействия скрытой установке вредоносного ПО могут быть различными. С увеличением разнообразия назначения и объемов данных популярными стали методы машинного обучения и непосредственно искусственные нейронные сети, которые, в частности, позволяют решать задачи распознавания образов, классификации и анализа скрытых закономерностей в данных [4].

Рассмотрим задачу обнаружения вредоносного ПО как задачу классификации, решение которой преследует цель повышения вероятности обнаружения вредоносных программ ОС Android.

Анализ существующих механизмов обнаружения вредоносных программ

В системе Android можно выделить два основных механизма обнаружения вредоносных программ: систему тестирования Google Bouncer и антивирусы.

Как правило, по умолчанию устройства под управлением ОС Android не поддерживают установку приложений из неизвестных источников, к числу которых относятся приложения, не подписанные сертификатом Google. Если разработчик приложения получил сертификат, то оно может быть опубликовано в магазине Google Play Store. Перед публикацией приложение проходит проверку системой Google Bouncer, являющейся антивирусным функционалом Google. Проверка заключается в тестировании приложения на наличие вредоносного кода посредством облачного сервиса, то есть приложение устанавливается на виртуальную машину и проводится динамический анализ. При обнаружении подозрительной активности приложение не проходит верификацию и не публикуется,

аккаунт разработчика при этом может быть заблокирован [5].

Другой подход к обнаружению вредоносных программ – использование антивирусов. Существующие для ОС Android антивирусы являются приложениями, а не компонентами ОС, что, собственно, и определяет их основной недостаток – изоляцию антивирусов от других приложений, что позволяет проводить только статический анализ [6].

Дополнительно к существующим подходам стоит отметить наличие механизма разрешений, суть которого в том, что, во-первых, ни одно приложение ОС Android по умолчанию не имеет разрешений на операции, способные повлиять на получение личных данных, ОС или другие приложения, а во-вторых, приложениям назначаются привилегии и только от пользователя зависит, давать ли доступ к данным тому или иному приложению.

В ОС Android до 6-й версии все разрешения выдаются при установке. Пользователю выводится список доступных приложению разрешений, и для продолжения установки он должен подтвердить, что разрешает приложению использовать перечисленные компоненты системы. В новых версиях Android выделена группа особых разрешений, называемых опасными, которые будут запрашиваться непосредственно перед использованием, тем самым не позволяя программе скрыто производить нежелательные действия на устройстве. Все разрешения, используемые приложением, должны быть указаны в манифесте [7].

Недостатки существующих механизмов защиты от вредоносных программ

Основным недостатком динамического анализа, на котором реализована система Google Bouncer, является то, что спустя какое-то время после проведенного анализа приложение может вновь запускать вредоносную программу. Существует множество методов определения завершения анализа на виртуальном устройстве: всевозможные датчики, камера, микрофон, Wi-Fi модуль, которыми снабжены современные смартфоны. Они позволяют обойти процедуру динамического анализа. Также известны методики обнаружения гипервизора, которые могут быть применены вредоносной программой без обращения к каким-либо датчикам или данным мобильного устройства. Помимо общих недостатков, связанных непосредственно с динамическим анализом, к минусам подхода, основанного на системе тестирования Google Bouncer, можно отнести то, что обязательную проверку проходят лишь приложения, распространяемые через Google Play. Для проверки приложений из других источников необходимы включенная опция валидации сторонних приложений и подключение к сети Интернет.

Для оценки эффективности мобильных антивирусов был проведен эксперимент, заключающийся в следующем:

– для вредоносной программы из открытого источника был определен показатель ее выявления на сервисе Virustotal, который составил 41 % из 56 %;

– программа декомпилировалась;

– в программу вносились различные изменения;

– программа заново компилировалась;

– на сервисе Virustotal оценивались новые показатели выявления.

Полученные результаты приведены в таблице 1. Эксперимент показал, что антивирусы не могут успешно обнаруживать программы, к коду которых была применена обфускация.

Таблица 1

Показатели выявления измененной вредоносной программы

Table 1

Indicators of detected malware change

Внесенное изменение	Показатель выявления, %
Программа без изменений	41/56
Переподпись приложения	29/56
Переименование пакета приложения	23/56
Изменения манифеста	22/56
Изменения ресурсов	25/56
Шифрование строк в байт-коде программы	13/56
Переименование классов в байт-коде	21/56
Переименование классов, методов и шифрование строк в байт-коде программы	8/56

Недостаток антивирусов кроется в ограничениях статического анализа приложений, применение которого требует наличия сигнатур, то есть новые вредоносные программы, для которых сигнатуры еще не выпущены, не могут быть обнаружены. К тому же к коду вредоносной программы может быть применена обфускация для усложнения анализа [8].

Основной недостаток механизма разрешений состоит в большом количестве разрешений, которые пользователь должен прочесть при установке приложения, при этом угрозы того или иного разрешения не всегда очевидны. Только стандартных разрешений в ОС Android порядка 300. Не решает этих проблем и появившаяся в 6-й версии ОС Android функция динамически запрашиваемых разрешений. Для поддержки приложений, написанных ранее, разработчики ОС Android оставили возможность получать все необходимые разрешения

при установке. Для этого нужно указать в манифесте, что программа разрабатывалась для версий Android, предшествующих 6-й.

Предлагаемое решение

Задачу обнаружения вредоносных программ ОС Android можно рассматривать как задачу классификации, для решения которой привлекаются инструменты нейронной сети.

Все приложения ОС Android в рамках поставленной задачи разделим на два класса: безопасные и вредоносные. Для классификации необходимо выделить признаки, по которым можно определить принадлежность программы к одному из классов. В качестве классификационных признаков могут быть использованы действия, которые программа производит на устройстве, а для получения признаков приложения ОС Android – манифест, байт-код приложения и динамический анализ.

Из манифеста приложения можно получить список разрешений приложения и список обрабатываемых широковещательных намерений.

По списку разрешений можно определить функции, которые программе разрешено выполнять на устройстве. Однако, как уже упоминалось ранее, список разрешений может быть очень большим, к тому же далеко необязательно использование приложением всех полученных разрешений. Из всего списка возможных разрешений необходимо выделить наиболее важные и более свойственные вредоносным программам. По списку обрабатываемых широковещательных намерений можно определить, какие события на устройстве отслеживает программа: подключение к Интернету, получение SMS-сообщения, входящий или исходящий вызов, включение устройства и другие. Получатели широковещательных намерений используются многими вредоносными приложениями, например, для перехвата SMS-сообщений, подтверждающих покупку с банковской карты, или для ожидания события подключения к Интернету для передачи украденных личных данных.

По байт-коду приложения можно определить, какие действия производит программа на устройстве. В качестве признаков, полученных из байт-кода, можно выделить вызовы к Android API и библиотекам ядра. Полученные в результате анализа байт-кода списки вызовов к API или библиотекам ядра достовернее списка разрешений, однако получение признаков из байт-кода может быть затруднено использованием динамической загрузки кода, механизмов рефлексии, обфускации. Динамическая загрузка кода позволяет скрыть почти весь функционал программы от анализа байт-кода, но сам факт того, что код приложения загружается из Интернета или зашифрованного файла, является подозрительным и может использоваться как признак для классификации.

Признаки могут быть получены и при динамическом анализе: при запуске исследуемой программы на реальном или виртуальном устройстве будут отслеживаться все действия программы, выполненные ею без участия пользователя.

Использование данных, полученных статическим и динамическим анализом одновременно, может устранить их недостатки, тем самым повысив вероятность обнаружения вредоносной программы. Таким образом, в сравнении с рассмотренными механизмами обнаружения вредоносных программ ОС Android предлагаемый подход имеет ряд преимуществ:

– способность обнаруживать ранее неизвестные вредоносные программы;
– отсутствие необходимости базы сигнатур;
– стойкость к обфускации.

Способность обнаруживать ранее неизвестные вредоносные программы достигается применением нейронных сетей, которые позволяют выявлять сложные зависимости между входными и выходными данными. При обучении на существующих программах нейронная сеть выявляет закономерности, свойственные приложениям каждого класса, и в результате способна верно определять, к какому из классов относится программа, отсутствующая в обучающей выборке.

Предлагаемый подход не требует сигнатур для анализа, поскольку все признаки, по которым классифицируются приложения, могут быть получены из приложения в результате проведенного анализа. Совместное применение статического и динамического анализа позволяет избавиться от некоторых их недостатков. Например, если вредоносная программа обнаружит, что ее анализируют, она может приостановить свою работу, не позволяя провести динамический анализ. Однако на работу статического анализатора такая способность программы не повлияет. Статический анализ может быть значительно затруднен обфускацией, но динамический анализ работает на уровень ниже в программном стеке ОС Android и обнаруживает реальные действия программы, которые невозможно скрыть обфускацией. Стойкость к обфускации также обеспечивается способностью нейронной сети получать верные результаты на искаженных или неполных входных данных.

Для оценки эффективности предлагаемого подхода создана его программная реализация, представляющий собой статический и динамический анализатор, а также классификатор на основе нейронной сети.

Статический анализатор распаковывает пакет приложения, анализирует байт-код и манифест приложения. К байт-коду приложения применяется декомпиляция, а к манифесту – десериализация. Для декомпиляции была использована программа smali.

Результатом работы статического анализатора являются списки

– функций Android API и библиотек ядра, которые могут быть вызваны из кода программы в процессе работы;

– разрешений приложения, которые запрашиваются при установке;

– широковещательных намерений, которые могут быть обработаны программой.

Динамический анализатор представляет собой модифицированную версию эмулятора ОС Android – droidbox, который позволяет отслеживать вызовы к Android API [7]. Приложение может взаимодействовать с устройством только через Android API, поэтому все действия программы будут обнаружены. Проверяемая программа запускается на виртуальном устройстве и работает в течение 5 минут. Во время работы программы отслеживались следующие действия:

– отправка данных в сеть;

– операции чтения и записи файлов на устройстве;

– использование DexClassLoader;

– отправка и чтение SMS-сообщений;

– чтение сохраненных на устройстве контактов;

– совершение звонков;

– использование встроенного криптопровайдера;

В качестве классификатора использовалась нейронная сеть с архитектурой многослойный персептрон с двумя скрытыми слоями. Обучение производилось методом обратного распространения ошибки [9, 10]. В качестве примеров приложений, относящихся к классу безопасных, были взяты 100 из 1 000 самых популярных приложений из магазина Google Play. Программы, относящиеся к классу вредоносных, были получены из Android Malware Genome Project.

Анализ результатов работы

Для оценки работы предлагаемого решения были проанализированы существующие программы для ОС Android. Анализ проводился на 25 вредоносных и на 50 безопасных программах. Полученные результаты отражены в таблице 2.

Таблица 2

Результаты анализа программ

Table 2

Results of program analysis

Характеристика результата	Количество
Проанализировано программ	75
Вредоносных программ	25
Безопасных программ	50
Верный результат классификации	64
Ложноотрицательные результаты	4
Ложноположительные результаты	3

В результате анализа было верно классифицировано 84 % вредоносных программ. Следует отметить, что проанализированные программы не использовались в обучении сети. Таким образом, проведенный анализ подтверждает способность предлагаемого решения верно классифицировать ранее неизвестные программы.

Для оценки способности предлагаемого решения обнаруживать вредоносные программы, к которым была применена обфускация, проанализированы использовавшиеся в эксперименте с антивирусами измененные версии вредоносной программы. Все экземпляры были верно определены как вредоносные. Результаты анализа, представленные в таблице 3, показывают способность предлагаемого решения обнаруживать обфусцированные версии вредоносных программ.

Таблица 3

Результаты анализа измененной вредоносной программы

Table 3

Results of the analysis of a changed malicious program

Внесенное изменение	Результат
Программа без изменений	+
Переподпись приложения	+
Переименование пакета приложения	+
Изменения манифеста	+
Изменения ресурсов	+
Шифрование строк в байт-коде	+
Переименование классов в байт-коде	+
Переименование классов, методов и шифрование строк в байт-коде	+

Заключение

Анализ информационных источников и проведенный эксперимент на программах, к коду которых была применена обфускация или у которых отсутствуют их сигнатуры, показал, что существующих механизмов обнаружения вредоносных программ ОС Android недостаточно.

Комплексное применение статического и динамического анализа программ ОС Android позволяет получить достаточное множество признаков, позволяющих с высокой долей вероятности безошибочно классифицировать исследуемые программы.

Задача классификации решена с применением аппарата нейронных сетей. Результаты показали, что все вредоносные программы, к байт-коду которых была применена обфускация, верно определены как вредоносные. Нейронная сеть верно классифицировала 84 % новых вредоносных программ.

Литература

1. Worldwide smartphone forecast update. 2016–2020, December 2016. URL: http://www.idc.com/getdoc.jsp?containerId=US42060116 (дата обращения: 27.01.2018).

2. Безопасность Android: взгляд внутрь. URL: http://www.securitylab.ru/blog/personal/Informacionnaya_bezopasnost_v_detalyah/325122.php (дата обращения: 18.01.2018).

3. Соглашение Google Play о распространении программных продуктов. URL: https://play.google.com/intl/ALL_ru/about/developer-distribution-agreement.html (дата обращения: 22.01.2018).

4. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. М.: Горячая линия–Телеком, 2013. 384 с.

5. Using DroidBox for dynamic malware analysis. URL: https://hackmag.com/uncategorized/droidbox-for-dynamic-malware-analysis/ (дата обращения: 03.01.2018).

6. Android Platform Architecture. URL: https://developer.android.com/guide/platform/index.html (дата обращения: 25.01.2018).

7. Android App permissions explained. URL: http://www.androidauthority.com/android-app-permissions-explained-642452/ (дата обращения: 10.01.2018).

8. Татарникова Т.М. Защищенные корпоративные сети. Раздел: Задачи по защите информации. СПб: Изд-во РГГМУ, 2012. 113 с.

9. Татарникова Т.М. Задача синтеза комплексной системы защиты информации в ГИС // Ученые записки РГГМУ. 2013. № 30. С. 204–211.

10. Тархов Д.А. Нейросетевые модели и алгоритмы. М.: Радиотехника, 2014. 349 с.

Software & Systems
DOI: 10.15827/0236-235X.123.543-547

Received 29.01.18
2018, vol. 31, no. 3, pp. 543–547

A neural network method for detecting malicious programs on the Android platform

T.M. Tatarnikova [1], *Dr.Sc. (Engineering), Associate Professor, Professor, tm-tatarn@yandex.ru*
A.M. Zhuravlev [1], *Graduate Student, asn93@mail.ru*

[1] *St. Petersburg State University of Aerospace Instrumentation, St. Petersburg, 190000, Russian Federation*

Abstract. Continuous growth in the number of malicious programs aimed at the Android operating system makes the problem of their detection very important. The lack of a centralized mechanism for distributing applications and effectiveness of existing solutions exacerbate the problem.

The paper demonstrates experimentally the lack of effectiveness of the existing mechanisms for detecting malicious programs in the Android operating system. The need for an automated solution that increases the probability of malicious program detection in the Android operating system including previously unknown modified and obfuscated versions of already known programs determined the purpose of the work.

The authors used the methods of neural network classification, static code analysis and analysis of program behavior in a virtual environment. The novelty of the proposed solution is the use of classification features obtained both by static code analysis and by analysis of program behavior on a virtual device. The proposed approach eliminates the shortcomings of existing solutions to the problem under consideration.

The ability of the proposed solution to detect malicious programs that were not used in neural network training, as well as obfuscated instances, has been experimentally proven.

The practical significance of the proposed solution is its use for building malware detection systems of the Android operating system, which can be applied in the Android application store.

Keywords: android platform, malware, antivirus, machine learning, neural networks, classification, program analysis.

References

1. *Worldwide Smartphone Forecast Update*. 2016–2020: December 2016. International Data Corporation (IDC). Available at: http://www.idc.com/getdoc.jsp?containerId=US42060116 (accessed January 27, 2018).

2. *Android Security: Look Inside*. Available at: http://www.securitylab.ru/blog/personal/Informacionnaya_bezopasnost_v_detalyah/325122.php (accessed January 18, 2018).

3. *Google Play Software Distribution Agreement*. Available at: https://play.google.com/intl/ALL_ru/about/developer-distribution-agreement.html (accessed January 22, 2018).

4. Rutkovskaya D., Pilinsky M., Rutkovsky L. *Neural Networks, Genetic Algorithms and Fuzzy Systems*. Moscow, Goryachaya liniya–Telekom Publ., 2013, 384 p.

5. *Using DroidBox for Dynamic Malware Analysis*. Available at: https://hackmag.com/uncategorized/droidbox-for-dynamic-malware-analysis/ (accessed January 3, 2018).

6. *Android Platform Architecture*. Available at: https://developer.android.com/guide/platform/index.html (accessed January 25, 2018).

7. *Android App Permissions Explained*. Available at: http://www.androidauthority.com/android-app-permissions-explained-642452/ (accessed January 10, 2018).

8. Tatarnikova T.M. *Protected Corporate Networks. Section: Information Security Tasks*. St. Petersburg, RGGMU Publ., 2012, 113 p.

9. Tatarnikova T.M. The task of synthesizing an integrated information security system in GIS. *Proc. of the Russian State Hydrometeorological Univ*. 2013, no. 30, pp. 204–211 (in Russ.).

10. Tarkhov D.A. *Neural Network Models and Algorithms*. Moscow, Radiotekhnika Publ., 2014, 349 p.