бителям продукции, технологические и производственные цепочки, производительность предприятий участников кластера, емкость складов и т. п.

По оценкам авторов размерность системы уравнений (1) может быть достигать 2000 и в, общем случае, и ее решение возможно только приближенными методами. В связи с этим требуют дополнительного исследования вопросы о необходимой вычислительной мощности компьютерной системы для вычисления решения, разрешимости системы уравнений, точности и устойчивости полученного решения.

ЛИТЕРАТУРА

1. Новикова И.В., Макуров Л.Г. Кластерная организация как институт развития в постиндустриальной экономике: методология анализа // Труды БГТУ. – 2019. – № 1. – С. 5-12.

2. И.В. Новикова, В.В. Смелова, Ю.А. Тимофеева, Д.В. Шиман. Концепция цифровой платформы инновационно-промышленного кластера // Импортозамещение, научно-техническая и экономическая безопасность : сб. ст. V Междунар. науч.-техн. конф. «Минские научные чтения – 2022», Минск, 7–9 декабря 2022 г.: в 3 т. – Минск : БГТУ, 2022. – Т. 2. – С. 3-7.

3. Ведута Е.Н. Межотраслевой-межсекторный баланс: механизм стратегического планирования экономики: Учебное пособие для вузов. – М.: Академический проект, 2020. – 239 с.

УДК 004.41.42

Доц. Н.А. Жиляк
(БГТУ, г. Минск)

СТАТИЧЕСКИЙ АНАЛИЗ КОДА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ КАК СРЕДСТВО ВЫЯВЛЕНИЯ ЕГО УЯЗВИМОСТЕЙ

Проблема информационной безопасности мобильных систем на сегодняшний момент особо актуальна. Это связано с масштабностью такого явления как мобильность: 6 млрд. абонентов сотовой сети по всему миру, около 3 млрд. мобильных устройств в сумме продано за прошлый год.

Рядовые пользователи не задумываются о безопасности конфиденциальной информации, которую злоумышленник может получить из их же мобильного устройства (личная переписка, финансовые данные, интеллектуальная собственность). Поэтому необходимо уделять проблематике безопасности мобильных систем особое внимание [1].

Можно привести следующую классификацию уровней угроз в отношении мобильных систем:

– физический (кража или непосредственное взаимодействие злоумышленника с мобильным устройством);

– аппаратный (закладки и уязвимости на аппаратном уровне, шпионские устройства, информация сотовых операторов);

– системный (закладки и уязвимости на уровне операционной системы);

– приложений (закладки и уязвимости в системных и сторонних приложениях, использование пользовательской информации, хранящейся на сервере не по назначению);

– социальный (методы социальной инженерии).

Можно также привести классификацию для средств защиты используемых в мобильных системах:

– аппаратные (шифрование с использованием специальных модулей встраиваемых в устройство, механическое отключение радиомодулей);

– системные (шифрование с использованием средств встроенных в операционную систему, системные механизмы авторизации и аутентификации);

– прикладные (антивирусы для мобильных устройств, анализаторы приложений, попадающих в маркеты, приложения для доступа к корпоративной сети);

– обучающие (материалы и курсы для повышения грамотности в сфере информационной безопасности).

Статический анализ кода – это проверка исходного кода приложения без реального выполнения на соответствие определенному набору правил. Такой вид анализа может использоваться для выявления ошибок и уязвимостей в исходном коде программного обеспечения. Программа выполняющая статический анализ называется статическим анализатором [2].

Анализ может проводиться либо над исходным кодом, либо над объектным (MSIL, байт-код). Управляемый код представляет функциональность методов приложения, закодированных в специальной бинарной форме, которая называется MSIL или Microsoft intermediate language (другие названия – Common Intermediate Language (CIL) или просто Intermediate Language (IL). Затем уже при выполнении сборки JIT-компилятор компилирует методы, закодированные в MSIL, в бинарный/машинный код текущей платформы, который затем собственно и выполняется. Статический анализ не гарантирует 100% выявления ошибок и уязвимостей. Возможны ложные (false positive) и

ложноотрицательные срабатывания (false negative) статического анализатора. По этой же причине статический анализатор в общем случае не предназначен для исправления найденных ошибок и уязвимостей. Он предупреждает программиста о подозрительных и потенциально проблемных участках кода [3].

Инструмент статического анализа сканирует код на наличие распространенных ошибок и уязвимостей, например утечек памяти и переполнения буфера. Кроме того, анализ помогает соблюдать стандарты написания кода.

Если вы придаете особое значение безопасности, специальные инструменты статического тестирования безопасности приложений (Static Application Security Testing, SAST) помогут вам выявить известные уязвимости. Статический анализ проводится на исходном коде, без запуска программы, поэтому его можно выполнять либо в самом начале CI/CD-пайплайна, либо прямо в IDE перед коммитом.

Как и другие автоматические тесты, статический анализ кода гарантирует последовательное выполнение проверок и позволяет быстро получить обратную связь по внесенным изменениям. Если инструменты статического анализа встроены в IDE, вы мгновенно получаете результаты проверок и можете устранять проблемы по ходу работы.

Однако статический анализ выявляет только те фрагменты кода, в которых нарушены правила программирования. Он не позволяет найти сразу все недочеты в исходном коде. Кроме того, результаты необходимо тщательно просматривать, поскольку возможны ложные ошибки.

С этой точки зрения статический анализ кода – важное дополнение к код-ревью: он показывает распространенные ошибки и освобождает время для более интересных задач, например, для общего проектирования.

Статический анализ кода – один из видов автоматических проверок, которые применяются для поддержания качества кода. Его следует использовать вместе с различными вариантами динамического анализа (в этом случае проверка кода на известные ошибки проводится при его выполнении) и автоматического тестирования.

Статический анализатор можно отнести к прикладному классу средств защиты используемых в мобильных системах. С помощью моделирования уязвимостей с последующим их внедрением в приложения, а также разработки алгоритма их определения можно построить статический анализатор, который будет выявлять уязвимости в приложениях для мобильных систем. Такой анализатор может ис-

пользовать механизм синтаксических деревьев с целью обеспечения более высокоуровневой работы с исходным кодом при его анализе на соответствие условиям алгоритма выявления определённой уязвимости. В материале рассмотрен статический анализ исходного кода программ для мобильных платформ как одно из средств обеспечения безопасности мобильных систем.

ЛИТЕРАТУРА

1. Drake, J. J. Android Hacker's Handbook. – Indianapolis: Wiley, 2020. – 576 p.

2. Bergman, N. Hacking Exposed: Mobile Security Secrets & Solutions. – NY: Mc Graw Hill, 2013. – 289 p.

3. Chess, B. Secure Programming with Static Analysis. – Boston: Addison-Wesley Professional, 2007. – 624 p.

УДК 003.26

Маг. А.Н. Николайчук; ст. преп. Е.А. Блинова
(БГТУ, г. Минск)

КОМБИНИРОВАННОЕ ПРИМЕНЕНИЕ ДВУХ СТЕГАНОГРАФИЧЕСКИХ МЕТОДОВ ДЛЯ РАЗМЕЩЕНИЯ ЦИФРОВОГО ВОДЯНОГО ЗНАКА В ФАЙЛАХ ФОРМАТА SVG

При хранении и передаче цифровых электронных документов важную роль играет обеспечение их безопасности. Одним из способов защиты информации является применение технологий цифрового водяного знака на основе стеганографического метода, суть которого заключается в скрытом размещении сообщения [1].

Ввиду уникальности свойств форматов цифровых документов необходимо разрабатывать специфические методы защиты для каждого из них. Однако для некоторых видов контейнеров могут рассматриваться методы, представляющие собой комбинацию различных подходов. Например, электронный текстовый документ формата SVG можно рассматривать как текст, как векторное изображение, как набор полей, содержащих метаинформацию и как контейнер, имеющий определенную структуру, что позволяет комбинировать классические подходы как текстовой, так и графической стеганографии [2−4].

Формат векторной графики SVG (Scalable Vector Graphics) позволяет легко манипулировать объектами, из которых состоит, благодаря своей структуре, описание файла осуществляется с помощью языка разметки XML. С его помощью можно создавать как примитив-