Russian Technological Journal.
2024;12(4):23–39

ISSN 2782-3210 (Print)
ISSN 2500-316X (Online)

Информационные системы. Информатика. Проблемы информационной безопасности

Information systems. Computer sciences. Issues of information security

УДК 004.31
https://doi.org/10.32362/2500-316X-2024-12-4-23-39
EDN DRCIUV

НАУЧНАЯ СТАТЬЯ

Выявление аппаратных уязвимостей цифровых устройств на основе систем сканирования и полунатурного моделирования

Е.Ф. Певцов,
Т.А. Деменкова [@],
А.О. Индришенок,
В.В. Филимонов

МИРЭА – Российский технологический университет, Москва, 119454 Россия
[@] Автор для переписки, e-mail: demenkova@mirea.ru

Резюме

Цели. Развитие вычислительной техники и информационных систем требует рассмотрения вопросов их безопасности, различных методов обнаружения аппаратных уязвимостей цифровых компонентов устройств и защиты от несанкционированного доступа. Важным аспектом данных проблем является исследование существующих методов на возможность и способность выявить аппаратные ошибки или произвести поиск ошибок на соответствующих моделях. Цель работы – разработка подходов, инструментов и технологии для обнаружения уязвимостей в аппаратном обеспечении на ранней стадии проектирования, создание методики их обнаружения и оценки риска, рекомендаций по обеспечению безопасности на всех этапах процесса разработки вычислительных систем.

Методы. Использованы методы полунатурного моделирования, сравнения и выявления аппаратных уязвимостей, стресс-тестирования для выявления уязвимостей.

Результаты. Предложены методы обнаружения и защиты от аппаратных уязвимостей, являющихся критически важным аспектом в обеспечении безопасности вычислительных систем. Для обнаружения уязвимостей в аппаратном обеспечении использованы методы сканирования портов, анализа протоколов связи и диагностики устройств. Определены возможные места нахождения аппаратных уязвимостей, их вариации, описаны атрибуты аппаратных уязвимостей и риски. Для обнаружения уязвимостей в аппаратном обеспечении на ранней стадии проектирования разработан специальный стенд полунатурного моделирования. Предложен алгоритм сканирования с использованием протокола Remote Bitbang, который позволяет передавать данные между *OpenOCD* и подключенным к отладочному порту устройством. На основе управления сканированием разработан метод верификации, реализующий сравнение поведенческой модели с эталоном. Приведены рекомендации по обеспечению безопасности на всех этапах процесса разработки вычислительных систем.

Выводы. В данной работе предложены новые технические решения для обнаружения уязвимостей в аппаратном обеспечении, основанные на таких методах, как сканирование системы на программируемой логической интегральной схеме, полунатурное моделирование, верификация по виртуальной модели, анализ протоколов связи и диагностика устройств. Применение разработанных алгоритмов и способов позволит разработчикам предпринять необходимые меры по устранению аппаратных уязвимостей и предотвращению возможных вредоносных воздействий на всех этапах процесса проектирования устройств вычислительной техники и информационных систем.

Ключевые слова: аппаратная уязвимость, цифровые компоненты, полунатурное моделирование, диагностика, сканирование, верификация

• **Поступила:** 30.08.2023 • **Доработана:** 13.01.2024 • **Принята к опубликованию:** 03.06.2024

Для цитирования: Певцов Е.Ф., Деменкова Т.А., Индришенок А.О., Филимонов В.В. Выявление аппаратных уязвимостей цифровых устройств на основе систем сканирования и полунатурного моделирования. *Russ. Technol. J*. 2024;12(4):23–39. https://doi.org/10.32362/2500-316X-2024-12-4-23-39

Прозрачность финансовой деятельности: Авторы не имеют финансовой заинтересованности в представленных материалах или методах.

Авторы заявляют об отсутствии конфликта интересов.

RESEARCH ARTICLE

Identification of digital device hardware vulnerabilities based on scanning systems and semi-natural modeling

Evgeniy F. Pevtsov,
Tatyana A. Demenkova [@],
Alexander O. Indrishenok,
Vladimir V. Filimonov

MIREA – Russian Technological University, Moscow, 119454 Russia
[@] Corresponding author, e-mail: demenkova@mirea.ru

Abstract

Objectives. The development of computer technology and information systems requires the consideration of issues of their security, various methods for detecting hardware vulnerabilities of digital device components, as well as protection against unauthorized access. An important aspect of this problem is to study existing methods for the possibility and ability to identify hardware errors or search for errors on the corresponding models. The aim of this work is to develop approaches, tools and technology for detecting vulnerabilities in hardware at an early design stage, and to create a methodology for their detection and risk assessment, leading to recommendations for ensuring security at all stages of the computer systems development process.

Methods. Methods of semi-natural modeling, comparison and identification of hardware vulnerabilities, and stress testing to identify vulnerabilities were used.

Results. Methods are proposed for detecting and protecting against hardware vulnerabilities: a critical aspect in ensuring the security of computer systems. In order to detect vulnerabilities in hardware, methods of port scanning, analysis of communication protocols and device diagnostics are used. The possible locations of hardware vulnerabilities and their variations are identified. The attributes of hardware vulnerabilities and risks are also described. In order to detect vulnerabilities in hardware at an early design stage, a special semi-natural simulation stand was developed. A scanning algorithm using the Remote Bitbang protocol is proposed to enable data to be transferred between *OpenOCD* and a device connected to the debug port. Based on scanning control, a verification method was developed to compare a behavioral model with a standard. Recommendations for ensuring security at all stages of the computer systems development process are provided.

Conclusions. This paper proposes new technical solutions for detecting vulnerabilities in hardware, based on methods such as FPGA system scanning, semi-natural modeling, virtual model verification, communication protocol analysis and device diagnostics. The use of the algorithms and methods thus developed will allow developers to take the necessary measures to eliminate hardware vulnerabilities and prevent possible harmful effects at all stages of the design process of computer devices and information systems.

Keywords: hardware vulnerability, digital components, half-life modeling, diagnostics, scanning, verification

• **Submitted:** 30.08.2023 • **Revised:** 13.01.2024 • **Accepted:** 03.06.2024

For citation: Pevtsov E.F., Demenkova T.A., Indrishenok A.O., Filimonov V.V. Identification of digital device hardware vulnerabilities based on scanning systems and semi-natural modeling. *Russ. Technol. J*. 2024;12(4):23–39. https://doi.org/10.32362/2500-316X-2024-12-4-23-39

Financial disclosure: The authors have no a financial or property interest in any material or method mentioned.

The authors declare no conflicts of interest.

ВВЕДЕНИЕ

Аппаратная уязвимость – ошибка в технической реализации аппаратуры, которая позволяет злоумышленникам получить доступ к системе или ее приложению. Современные аппаратные уязвимости могут иметь серьезные последствия для безопасности компьютерных систем, включая компрометацию конфиденциальных данных, нарушение работоспособности систем и потенциальную угрозу для жизни и здоровья людей в случае использования уязвимостей в медицинском оборудовании или автомобильной технике[1].

Аппаратные уязвимости могут быть различными и включают в себя ошибки в различных компонентах компьютера: процессоре, микросхемах, памяти, аппаратных модулях и драйверах. Также они содержатся в других аппаратных сложно-функциональных блоках (СФ-блоках). Аппаратные уязвимости могут проявляться в результате взаимодействия с программным обеспечением (ПО), в частности с драйверами, базами данных и другими приложениями, содержащими ошибки или зловредный код.

Проверка на наличие уязвимостей является сложным процессом и требует участия специалистов в области безопасности информации[2]. Определить аппаратную уязвимость можно с помощью сканирования исследуемой системы для нахождения проблемных мест и их последующего устранения [1]. Одним из вариантов поиска уязвимостей в существующих вычислительных устройствах является использование специализированных программных средств и инструментов. Например, существует множество утилит и приложений для поиска и анализа уязвимостей, которые могут быть использованы для проверки безопасности сетевых устройств и др. Или же может быть применен комплексный подход сравнения моделей в виде связки *OpenOCD*[3] и программной модели устройства, полученной с помощью *Verilator* [2] из языков описания аппаратуры[4].

Для поиска уязвимостей также применяют методы анализа и обратного инжиниринга. Эти методы могут использоваться для поиска уязвимостей в программно-аппаратных комплексах, операционных системах, драйверах устройств и других вычислительных системах [3].

Каждая уязвимость имеет определенные атрибуты, которые можно использовать для ее обнаружения. В статье описываются атрибуты аппаратных уязвимостей, методы их обнаружения и оценки риска. Атрибуты аппаратных уязвимостей рассмотрены в соответствии с принятой таксонометрией, предложенной в ряде работ классификацией по категориям идентификации аппаратных уязвимостей и оценки их риска или серьезности [4]. В этой таксонометрии выделены четыре многопараметрических признака (вектора), соответствующие идентификации аппаратных уязвимостей, идентификации обнаружения аппаратных уязвимостей, риску или

[1] Zantout S. *Hardware Trojan Detection in FPGA through Side-Channel Power Analysis and Machine Learning*. MSc Thesis. University of California, Irvine. 2018. https://escholarship.org/uc/item/7hk8x6rb. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[2] Oberg J. *Testing Hardware Security Properties and Identifying Timing Channels*. UC San Diego. 2014. P. 5–38. https://escholarship.org/uc/item/8b530988. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[3] Документация по OpenOCD RISC-V Debug Configuration Commands [Documentation regarding OpenOCD RISC-V Debug Configuration Commands]. https://openocd.org/doc/html/Architecture-and-Core-Commands.html#RISC_002dV-Debug-Configuration-Commands. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[4] Андрианов А.В. *Реализация возможности пошаговой отладки при отладке тестовых сценариев на модели СБИС СнК*. https://www.module.ru/uploads/media/1534156062-2018-833a272aac.pdf. Дата обращения 15.05.2023. [Andrianov A.V. *Realization of possibility of step-by-step debugging at debugging of test scenarios on VLSI SonC model*. https://www.module.ru/uploads/media/1534156062-2018-833a272aac.pdf (in Russ.). Accessed May 15, 2023.]

серьезности аппаратных уязвимостей и эффективности обнаружения аппаратных уязвимостей.

ПОСТАНОВКА ЗАДАЧИ. ТАКСОНОМЕТРИЯ АППАРАТНЫХ ЗАКЛАДОК

Каждая аппаратная уязвимость имеет ряд атрибутов, поэтому были разработаны методы, позволяющие выполнять их обнаружение с отдельными или всеми атрибутами. Информация о векторах и атрибутах в дальнейшем необходима для подбора методов выявления уязвимостей.

Ранжирование атрибутов в каждой категории может осуществляться на основе их важности, уникальности, весовых коэффициентов или других критериев, определенных в рамках конкретного проекта или задачи по обнаружению троянов. Под аппаратным трояном понимается вредоносный модуль, встроенный в интегральную схему, либо вредоносная модификация схемы для изменения ее работы или добавления дополнительного функционала, например, для организации канала утечки информации.

Вектор ИТ (идентификация трояна) включает атрибуты, которые используются для идентификации троянов, например, размер трояна, расположение в схеме, механизмы защиты, используемые трояном, и другие характеристики, которые могут быть уникальными для конкретного трояна.

Вектор ИОТ (идентификация обнаружения трояна) включает атрибуты, которые могут быть обнаружены при использовании определенного метода обнаружения трояна, например, изменения в сигнатурах трояна, аномалии в поведении трояна, особенности атаки, используемые трояном, и другие признаки, которые могут быть обнаружены при применении конкретного метода.

Вектор СТ (риск или серьезность трояна) включает атрибуты, которые оценивают риск или серьезность трояна, например, возможность трояна причинить вред, степень скрытности трояна, возможность его распространения, потенциальные последствия его действий и другие факторы, которые могут влиять на серьезность трояна.

Вектор ЭОТ (эффективность обнаружения трояна) включает атрибуты, которые оценивают эффективность конкретного метода обнаружения трояна, например, точность обнаружения, ложноположительные и ложноотрицательные срабатывания метода, скорость обнаружения, сложность реализации метода и другие факторы, которые могут влиять на эффективность метода.

Эти векторы могут быть использованы для оценки троянов и выбора наиболее эффективного метода их обнаружения, идентификации и оценки риска или серьезности.

Комбинация значений векторов ИТ, ИОТ, СТ и ЭОТ может быть использована для выбора наиболее эффективного метода обнаружения трояна на основе определенных критериев и требований проекта или задачи. Например, метод с высокими значениями векторов ИТ и ЭОТ, указывающими на высокую способность обнаружения трояна и эффективность метода, может быть предпочтительнее, чем метод с более низкими значениями в этих векторах. Атрибуты в векторе СТ могут также использоваться для оценки степени серьезности трояна и его приоритета при выборе метода обнаружения.

Атрибуты в каждой категории используются для идентификации троянов и оценки их риска или серьезности. Они также могут быть использованы для определения методов обнаружения троянов и оценки их эффективности. Каждому трояну присваиваются два вектора: ИТ и СТ. Первый идентифицирует соответствующие атрибуты, а второй представляет атрибуты с точки зрения их риска или серьезности. Каждый метод обнаружения трояна также имеет два вектора: ИОТ и ЭОТ. ИОТ идентифицирует атрибуты, которые могут быть обнаружены, а ЭОТ представляет атрибуты с точки зрения эффективности метода. Таким образом, существует четыре вектора соответствующие: 1) ИТ, 2) ИОТ, 3) СТ, 4) ЭОТ [5].

Предложенный подход, основанный на ранжировании атрибутов троянов в каждой категории и использовании векторов ИТ, СТ, ИОТ и ЭОТ, может быть полезным инструментом для идентификации и оценки риска троянов, а также для выбора эффективных методов обнаружения при разработке более надежных систем безопасности и защиты от атак.

Однако следует отметить, что предлагаемая система ранжирования атрибутов троянов может быть ограничена, поскольку новые виды троянов и их атрибуты могут возникать в результате развития технологий и методов атаки. Поэтому необходимо постоянно обновлять и дополнять базу данных атрибутов троянов и методов их обнаружения для более эффективной защиты от них.

Обнаружение уязвимостей – это только первый шаг, необходимо предпринять меры по устранению этих уязвимостей и предотвращению возможных атак. В этом процессе важную роль могут играть инструменты, такие как системы автоматизированного проектирования (САПР), которые позволяют верифицировать устройства и обнаруживать уязвимости на ранней стадии проектирования. Типичным примером такой системы является комплекс инструментов САПР компании Cadence Design Systems [6]. В целом, безопасность является критически важным аспектом в разработке вычислительных систем, обнаружение и защита от аппаратных уязвимостей

должны учитываться на всех этапах процесса разработки и эксплуатации.

Помимо осуществляемой функции уязвимость можно модернизировать с целью удаленного взаимодействия, для чего необходимо запрограммировать возможность доступа к сетевой карте и выхода в сеть. Кроме этого, можно настроить закладку не только на порчу выходных данных в данный момент времени, но и на изменение уже записанных данных. Для этого закладка должна обладать выходом к хранилищам памяти и либо возможностью выявления определенных данных, либо возможностью изменения всех данных на определенную величину. Написание и модернизация закладки ограничиваются лишь навыками самого злоумышленника и размерами закладки [7–9].

ПРИМЕР РЕАЛИЗАЦИИ АППАРАТНОЙ ЗАКЛАДКИ

Важным аспектом методики выявления аппаратных закладок является процедура создания возможных примеров. Обычно основная трудность возникает в том, что именно надо показать и каким образом представить. Приведенный пример может быть полезен разработчикам защитных средств вычислительной техники при проектировании в части выявления возможных уязвимостей.

Рассмотрим аппаратные закладки, нарушающие корректность вычислений в ядрах аппаратного ускорителя математических вычислений. При сложении вещественных чисел на сумматоре, умножителе или делителе можно составить аппаратную закладку, нарушающую корректность вычислений. Внедрим закладку в алгоритм сложения и в код сумматора при помощи отдельного модуля и произведем синтез полученного вычислительного блока.

Модуль выполняет следующую процедуру: при поступлении в сумматор определенного числа закладка срабатывает и заменяет это число на то, которое нужно злоумышленнику, после чего передает это число непосредственно на суммирование. В случае, если закладка не получила необходимого ей для перехода числа, то она остается неактивной. Таким образом, закладка осуществляет порчу выходных данных, выдавая при этом свое появление в конкретный момент поступления на вход определенного числа. На рис. 1 представлен алгоритм работы блока сумматора для отработки навыков внедрения и выявления закладок в наиболее распространенных узлах цифровых систем вычислительной техники.

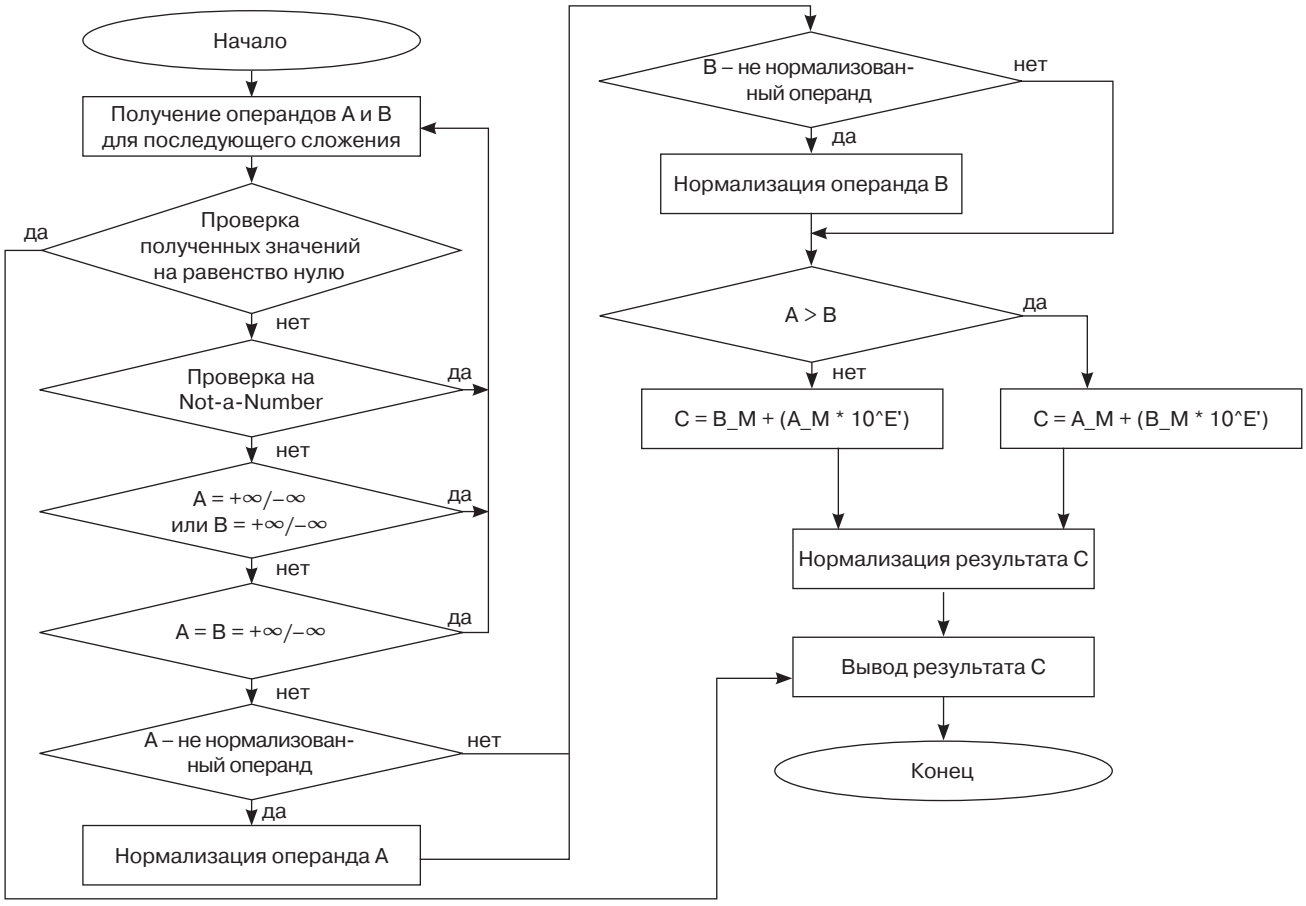

Рис. 1. Алгоритм работы сумматора

Аппаратная закладка при наличии дробной части в результате будет менять ее значение, изменяя точность вычисления. Наличие дробной части в числе в закодированном 32-битном формате стандарта IEEE 754[5] можно обнаружить по наличию единиц в разрядах

$$s + e + \exp + 1 \le n, \quad (1)$$

где s – количество бит, выделяемых под знак числа; e – количество бит, выделяемых под смещенную экспоненту числа; exp – экспонента числа; n – общее количество бит.

В 32-разрядном представлении (1) можно записать как:

$$10 + \exp \le 32. \quad (2)$$

Если число имеет дробную часть, то инвертируем разряд $s + e + \exp + 2$. Иными словами, число $5.5_{10} = 101.1\mathbf{0}0_2$ после инвертирования данного разряда станет равным $5.75_{10} = 101.1\mathbf{1}0_2$.

Блок-схема алгоритма аппаратной закладки приведена на рис. 2 [10, 11].

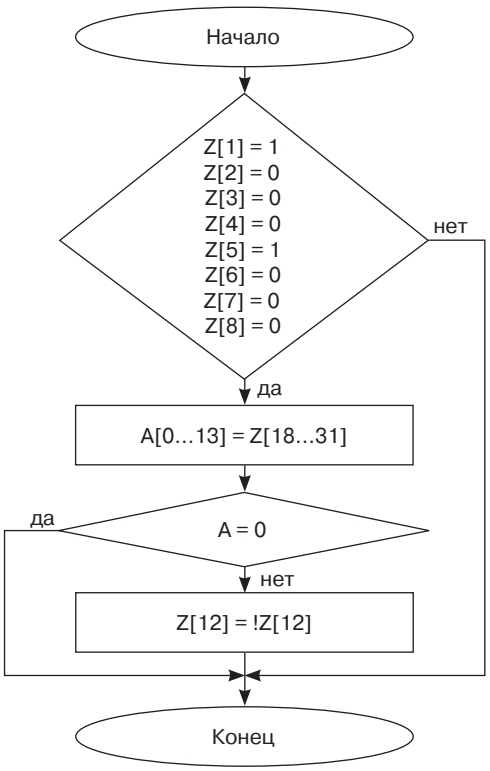

Рис. 2. Блок-схема алгоритма аппаратной закладки

Чтобы данное событие, при котором срабатывает аппаратная закладка, было менее частым, возьмем только числа, у которых, помимо наличия дробной части, значения будут приниматься только в пределах $512 \le x < 1024$. Условие на выполнение данного условия можно получить из значения экспоненты двоичного числа. Если экспонента двоичного числа равна $9_{10} = 1001_2$, что соответствует смещенной экспоненте 10001000_2, то данное число лежит в интервале $512 \le x < 1024$.

В результате данный аппаратный троян будет изменять результат на сумматоре, только если этот результат принадлежит интервалу $512 \le x < 1024$ и имеет дробную часть.

Так как мы определили порядок требуемых для искажения чисел, равный 9_{10}, то ранее описанный этап нахождения нужного разряда в числе на проверку, имеет ли данное число дробную часть, более не требуется. Любые числа 32-битного формата IEEE 754, лежащие в интервале $512 \le x < 1024$ и имеющие единицу в любом из $18 \le n \le 31$ разрядов, будут иметь дробную часть согласно формулам (1) и (2):

$$1 + 8 + 9 + 1 = 19 \le n.$$

Закодированная дробная часть числа будет расположена в интервале с 19 разряда по 32 разряд, а закодированная целая часть числа находится в разрядах с 9 по 17. Следовательно, для проверки, имеет ли число дробную часть, требуется найти хотя бы одну единицу в разрядах 18–31.

С шины результата сумматора считывается значение полученного закодированного числа в регистр Z аппаратного трояна. Если смещенная экспонента числа равна 10001000_2, то в регистр A аппаратной закладки записывается остаток мантиссы закодированного числа. Далее, если данный регистр A является не пустым, то в регистре Z инвертируется значение z 12 разряда, что соответствует второму разряду дробной части числа, в результате чего число либо увеличивается на 0.25, если в данном разряде находится 0, либо уменьшается на 0.25, если в данном разряде находится 1. Разработанный аппаратный троян может быть встроен в ядро сумматора вещественных чисел аппаратного ускорителя математических вычислений. Описание разрабатываемой закладки выполняется на языке Verilog [12].

После успешной разработки аппаратного трояна надо выполнить верификацию. Требуется определить, правильно ли работает аппаратная закладка в зависимости от заданных условий. Для данной проверки определим несколько возможных событий. Каждое из событий должно быть проверено как с наличием аппаратного трояна, так и при его отсутствии.

[5] 2754-2019 IEEE Standard for Floating-Point Arithmetic. July 22, 2019. Electronic ISBN 978-1-5044-5924-2. https://ieeexplore.ieee.org/document/8766229. Дата обращения 15.05.2023. / Accessed May 15, 2023.

1. Результат сумматора принадлежит интервалу $512 \le x < 1024$.
   а) результат имеет дробную часть;
   б) результат не имеет дробной части.
2. Результат сумматора не принадлежит интервалу $512 \le x < 1024$.
   а) результат меньше 512 и имеет дробную часть;
   б) результат меньше 512 и не имеет дробной части;
   в) результат больше или равен 1024 и имеет дробную часть;
   г) результат больше или равен 1024 и не имеет дробной части.
3. Результат сумматора является исключительным числом.
   а) результат равен 0;
   б) результат равен $+\infty$;
   в) результат равен $-\infty$.

Такую аппаратную уязвимость можно обнаружить, применяя следующие способы.

1. **Анализ аномального поведения**. Наличие закладки может привести к аномальному поведению математического блока (FPU, floating-point unit), такому как неожиданные ошибки вычислений, неправильные результаты, или необычная активность во время выполнения математических операций.
2. **Применение теста производительности Linpack** [13]. Аппаратную закладку, введенную в FPU, сложно обнаружить с помощью этого теста, поскольку его объем незначительно влияет на производительность. Однако этот метод окажется эффективным, если имеется возможность сравнить два аппаратных блока – с закладкой и без нее. Одна модель имеет намеренно внедренную аппаратную уязвимость и загружена в программируемую логическую интегральную схему (ПЛИС), а другая модель является поведенческой и реализуется с некоторыми упрощениями в симуляторе на компьютере при помощи специальных инструментов САПР для интегральных схем, например, инструментов Cadence Design Systems.
3. **Проверка набора инструкций**, выполняемых внутри процессора с использованием интерфейса JTAG (Joint Test Action Group)[6] и системы сканирования на основе протокола Remote Bitbang. В частности, аудит микрокода FPU на наличие необычных или подозрительных инструкций, которые могут указывать на наличие аппаратной закладки.

   Протокол Remote Bitbang — это протокол, используемый в *OpenOCD* (открытая программа для отладки встраиваемых систем), который позволяет передавать данные между *OpenOCD* и устройством, которое подключено к отладочному порту. Протокол Remote Bitbang используется для управления вводом-выводом и другими интерфейсами устройства.

   В протоколе Remote Bitbang используется асинхронная передача данных, которая представляется в виде простого формата сообщений. Команды, отправляемые по этому протоколу, имеют два типа: команды для записи и команды для чтения. Команды для записи позволяют управлять выводами, передавать данные и устанавливать режимы работы интерфейсов, а команды для чтения позволяют получать данные от устройства через вводы.

   Протокол Remote Bitbang используется в сочетании с такими протоколами как JTAG для обеспечения полноценной отладки и программирования встраиваемых систем. Этот протокол может быть использован для сканирования регистров на удаленном устройстве, подключенном к *OpenOCD*, и в процессах автоматизации тестирования, диагностики неисправностей и программирования устройств в производственных условиях [14–16].
4. **Использование специализированных инструментов** (анализаторы микрокода и специализированные стенды для обнаружения аппаратных закладок), которые могут быть полезны в выявлении таких угроз. Однако для их использования требуются специальные знания и опыт.

СТЕНД ПОЛУНАТУРНОГО МОДЕЛИРОВАНИЯ

При разработке стенда, предназначенного для выявления аппаратных уязвимостей, принималось во внимание, что он должен обеспечить достоверность полученных результатов и выполнение следующих функций:

1. Возможность тестирования различных типов аппаратного обеспечения, включая процессоры, чипсеты, контроллеры устройств ввода-вывода, сетевые карты и другие компоненты.
2. Возможность использования различных инструментов для обнаружения уязвимостей в аппаратном обеспечении, включая сканеры уязвимостей, дебаггеры, эмуляторы и другие средства.
3. Возможность проведения экспериментов в различных сценариях, включая атаки на аппаратное обеспечение с помощью вредоносных программ, манипуляции с памятью и периферийными устройствами, а также эксплойты уязвимостей.
4. Обеспечение безопасности и защиты данных во время проведения экспериментов, включая

[6] https://ru.wikipedia.org/wiki/JTAG (in Russ.). Дата обращения 15.05.2023. / Accessed May 15, 2023.

Рис. 3. Структура стенда для проведения экспериментов по выявлению аппаратных уязвимостей. ПЗУ – постоянное запоминающее устройство; SDRAM, synchronous dynamic random access memory – синхронная динамическая память с произвольным доступом; МК stm32 – микроконтроллер STM32

защиту от утечки конфиденциальной информации и обеспечение конфиденциальности результатов тестирования.

5. Обеспечение возможности воспроизведения результатов тестирования и документирования всех шагов эксперимента, включая средства автоматизации тестирования и инструменты для анализа результатов.

Структурная схема стенда приведена на рис. 3.

Особенностью данной структуры является специальная структура ПЛИС, позволяющая проводить повторную проверку прошивки для гарантирования отсутствия изменений в схеме CPLD[7]. Также есть возможность использовать встроенные ресурсы ПЛИС для базовой проверки прошивки с использованием встроенных средств контроля.

Верификация Verilog-кода, который используется для формирования файла прошивки (bitstream), может быть выполнена с использованием различных методов и инструментов, таких как симуляция, формальная верификация и эмуляция на аппаратном обеспечении.

1. **Симуляция**. После создания тестовых векторов, которые представляют различные сценарии загрузки bitstream в ПЛИС, следует использовать симулятор Verilog для выполнения симуляции этих тестовых векторов. В результате можно анализировать значения сигналов внутри схемы сравнения bitstream и сравнивать их с ожидаемыми значениями. Это позволит обнаружить ошибки и проверить правильность работы методов сравнения bitstream.
2. **Формальная верификация** – метод проверки корректности Verilog-кода на основе математических алгоритмов. Можно использовать инструменты формальной верификации, например, model checking или equivalence checking, для проверки корректности работы методов сравнения bitstream. Это может включать проверку правильности логики сравнения, детектирование потенциальных ошибок и нахождение неожиданных путей выполнения.
3. **Эмуляция на аппаратном обеспечении**. Загрузка bitstream в ПЛИС и запуск его на физическом устройстве. Затем можно использовать тестовые векторы и физические сигналы для проверки корректности работы методов сравнения bitstream в реальном времени. Это может помочь выявить возможные проблемы при работе на реальном аппарате.
4. **Ручная проверка**. Можно внимательно анализировать код методов сравнения bitstream и проводить ручную проверку на соответствие требованиям и ожидаемому поведению. Это может включать анализ логики, проверку граничных условий и тестирование различных сценариев загрузки bitstream.

Другой особенностью архитектуры стенда полунатурного моделирования является возможность загрузки во внутреннюю память через внутренний интерфейс реконфигурирования ПЛИС. Если такой интерфейс отсутствует, то можно использовать стандартный внешний интерфейс реконфигурации. Для того чтобы избежать проблем при проверке и загрузке в память из буфера FIFO[8], необходимо размещать блоки проверки в строго отведенных ячейках ПЛИС при написании конфигурации и раскладке на кристалле. Кроме того, этот модуль имеет многоступенчатую проверку, которая позволяет обнаруживать угрозы, сообщать о них и устранять возможные негативные последствия. Отчет о результатах проверки сохраняется и может быть использован для дальнейшего анализа и улучшения работы модуля.

Поскольку архитектуры ПЛИС в зависимости от модели и производителя могут отличаться, данная

[7] Complex programmable logic device – сложное программируемое логическое устройство.

[8] FIFO – first in, first out.

структура позволяет адаптировать алгоритмы проверки к разным архитектурам и производителям. По сравнению с ближайшими аналогами данный модуль является более актуальным, поскольку имеет возможность подключения к различным внешним устройствам, что обеспечивает его универсальность и возможность применения в качестве ведущего устройства в системах коммутации информации, а также в системах с высокопроизводительными процессорами [17, 18].

Перед началом сканирования необходимо выбрать режимы работы устройства, которые позволят определить возможные уязвимости в системе передачи данных по интерфейсу RS-485. Например, можно настроить устройство на отправку злонамеренных команд или на перехват и анализ данных, передаваемых по сети. При этом необходимо учитывать, что такие действия могут привести к нарушению работы устройства или сети в целом, поэтому проведение экспериментов необходимо проводить в контролируемой среде и с предварительным согласованием с ответственными за работу системы [1, 19].

Пример реализации стенда полунатурного моделирования приведен на рис. 4.

Рис. 4. Реализация стенда полунатурного моделирования

В ПЛИС передачи данных загружается конфигурация в соответствии с целевой моделью, которая описывает ожидаемые результаты работы устройства. После этого начинается процесс получения показаний с внешних интерфейсов исследуемого блока, таких как RS-485 или Ethernet[9], и их сравнение с целевой моделью. В случае если результаты не соответствуют ожидаемым, формируется ошибка и заносится в отчет, а индикатор на устройстве сигнализирует о возникновении ошибки[10].

При подключении стенда и отладки прошивки ПЛИС для получения необходимых данных для произведения сравнения с целевой моделью, развернутой на компьютере, важно обратить внимание на то, что схема подключения программатора JTAG к ПЛИС может отличаться в зависимости от конкретной модели программатора и ПЛИС, а также от задачи, которую необходимо решить при помощи данного подключения[11, 12, 13]. В этом случае при необходимости подключения дополнительных периферийных устройств к ПЛИС, например, для отладки системы, содержащей микроконтроллер, может потребоваться подключение специального отладочного модуля.

ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ

Тест стенда полунатурного моделирования выполнен на основе проверки проекта процессора, например, RISC-V[14], в поведенческое описание которого на языке Verilog введен код намеренной аппаратной уязвимости. Данная уязвимость может быть представлена в виде модифицированного СФ-блока, готового к встраиванию в систему с архитектурой RISC-V для проведения экспериментов по выявлению аппаратной уязвимости[15] [20].

[9] Андрианов А.В. *Реализация возможности пошаговой отладки при отладке тестовых сценариев на модели СБИС СнК*. https://www.module.ru/uploads/media/1534156062-2018-833a272aac.pdf. Дата обращения 15.05.2023. [Andrianov A.V. *Realization of possibility of step-by-step debugging at debugging of test scenarios on VLSI SonC model*. https://www.module.ru/uploads/media/1534156062-2018-833a272aac.pdf (in Russ.). Accessed May 15, 2023.]

[10] Fern N.C. *Verification Techniques for Hardware Security*: Ph.D. Thesis (Comput.). USA: UC Santa Barbara; 2016. P. 10–25. https://escholarship.org/uc/item/2ch6f44s. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[11] Документация Cadence [Cadence documentation]. https://www.cadence.com/content/cadence-www/global/en_US/home/support/documentation.html. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[12] Yang P. *Assessing VeSFET Monolithic 3D Technology in Physical Design, Dynamic Reconfigurable Computing, and Hardware Security*: Ph.D. Thesis (Comput.). USA: UC Santa Barbara; 2017. P. 62–81. https://escholarship.org/uc/item/5s9833kz. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[13] Farinholt B.R. *Understanding the Remote Access Trojan malware ecosystem through the lens of the infamous DarkComet RAT*: Ph.D. Thesis (Comput.). Sci. USA: UC San Diego; 2019. P. 17–29. https://escholarship.org/uc/item/3vv544n5. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[14] RISC-V – расширяемая открытая и свободная система команд и процессорная архитектура на основе концепции RISC (reduced instruction set computer), предназначенная для создания процессоров/микроконтроллеров и разработки ПО. [RISC-V is an extensible open and free instruction system and processor architecture based on the RISC (reduced instruction set computer) concept for processor/microcontroller design and software development.]

[15] Li C. *Securing Computer Systems Through Cyber Attack Detection at the Hardware Level*: Ph.D. Thesis (Comput.). USA: UC Irvine; 2020. P. 13–26. https://escholarship.org/uc/item/8vr8f0dq. Дата обращения 15.05.2023. / Accessed May 15, 2023.

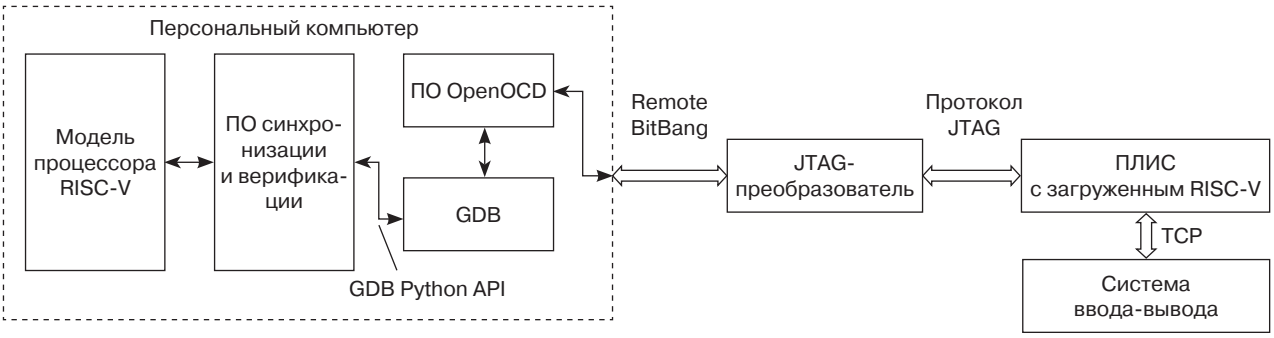

Рис. 5. Модель для поиска уязвимостей в основанной на процессоре RISC-V системе. API, application programming interface – программный интерфейс приложения; TCP, transmission control protocol – протокол передачи данных; GDB, Gnu DeBugger – отладчик с открытым кодом

После внедрения модифицированного аппаратного СФ-блока необходимо провести сравнение работы и попытаться выявить узел с уязвимостью сначала с использованием ПО на выявление ошибок. Затем с применением анализа виртуальной модели провести анализ на ПЛИС с использованием связки протоколов Remote Bitbang и JTAG, а также ПО *OpenOCD*, после чего провести сравнение результатов и выявить возможные отклонения от созданной поведенческой модели устройства. Общая концепция идеи сканирования представлена на рис. 5.

Под поведенческой моделью устройства понимается абстрактная модель, описывающая функциональное поведение устройства и его взаимодействие с другими компонентами системы. Она описывает, как конкретный блок должен вести себя при определенных входных сигналах и условиях без уточнения внутренней реализации устройства, и как это поведение отражается на общей картине взаимодействия устройства с внешними источниками информации.

Предполагается, что для проверки следует использовать описание на языке UML (unified modeling language), который применяется для описания поведения устройства в целом, но возможны и другие варианты.

Для проверки кода Verilog на наличие уязвимостей или аппаратных закладок можно использовать системы автоматической проверки безопасности. Такие системы анализируют код на наличие ошибок и ищут возможные уязвимости. Для анализа и проверки кода Verilog на уязвимости также можно использовать сторонние инструменты, такие как *Verilator* и *VeriSim*[16].

Verilator – это транслятор с открытым исходным кодом из Verilog в Си/C++, генерирующий файлы для профилирования и отладки. Он позволяет пользователям проанализировать код на наличие аппаратных закладок и уязвимостей. Этот инструмент можно использовать для автоматической проверки кода Verilog на ошибки.

VeriSim – другой инструмент для анализа и проверки кода Verilog. Он позволяет пользователям проанализировать код, используя симуляцию и профилирование. Также он позволяет идентифицировать аппаратные закладки и уязвимости в коде Verilog.

Альтернативными инструментами для анализа и проверки кода Verilog на наличие аппаратных закладок и уязвимостей могут являться *Vivado HLS*[17], *Synopsys VCS*[18] и *Mentor Questa*[19].

Для поиска уязвимостей в коде программы на языке SystemC можно использовать автоматические инструменты поиска уязвимостей, такие как *SonarQube*[20] и *Coverity*[21]. Эти инструменты позволяют автоматизированно просканировать исходный код на предмет уязвимостей. В случае, если уязвимости или аппаратные закладки не будут найдены, существует риск того, что приложение будет подвержено атакам со стороны злоумышленников, которые могут воспользоваться уязвимостью для внесения

[16] Wei S. *Minimizing Leakage Energy in FPGAs Using Intentional Post-Silicon Device Aging*: Master Sci. Thesis. USA: UC Los Angeles; 2013. P. 16–35. https://escholarship.org/uc/item/75h4m6qb. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[17] https://docs.xilinx.com/r/en-US/ug949-vivado-design-methodology/Vivado-Design-Suite-User-and-Reference-Guides. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[18] https://users.ece.utexas.edu/~patt/10s.382N/handouts/vcs.pdf. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[19] https://www.orcada.ru/product/mentor-graphics/proektirovanie-zakaznyh-ims/products_106.html (in Russ.). Дата обращения 15.05.2023. / Accessed May 15, 2023.

[20] https://www.sonarsource.com/products/sonarqube/. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[21] https://devguide.python.org/development-tools/coverity/. Дата обращения 15.05.2023. / Accessed May 15, 2023.

```
set(MCU_RISCV RISCV)
set(START_FILE startup.s) # стартовый файл s

add_compile_options(-Wall -Wextra)
add_compile_options(-O2 -ggdb)
add_link_options(-mthumb -mfpu=fpv4-sp-d16 -mfloat-abi=hard
                 -T${RISCV_LDSCRIPT} --specs=nosys.specs --specs=nano.spec

# set the project name
project(Test_firs_prj VERSION 0.1)

include_directories(
                    "${PROJECT_BINARY_DIR}"
                    "${PROJECT_SOURCE_DIR}/inc"
                    "${PROJECT_SOURCE_DIR}/library/CMSIS"
                     )

set_property(SOURCE ${START_FILE} PROPERTY LANGUAGE C)

add_subdirectory(library/CMSIS)

configure_file(inc/version.h.in inc/version.h)

list(APPEND TARGET_SOURCE ${PROJECT_SOURCE_DIR}/inc/main.h)
list(APPEND TARGET_SOURCE ${PROJECT_SOURCE_DIR}/src/main.c)
```

Рис. 6. Пример системы сборки программных тестов

в код нежелательных изменений или получения доступа к данным[22, 23, 24, 25, 26] [21, 22].

Для выполнения синхронизации между виртуальным процессором и реальным процессором, которые проходят отладку, можно использовать возможности *OpenOCD*. Необходимо подключить *OpenOCD* к отладочному интерфейсу реального процессора. Это можно сделать, например, через интерфейс JTAG. Затем необходимо настроить *OpenOCD* на работу с виртуальным процессором, который проходит отладку, например, с процессором, эмулируемым в *QEMU* [23] или посредством Functional Safety Simulator[27].

Далее необходимо выполнить синхронизацию между виртуальным и реальным процессором. Это можно сделать с помощью команды «resume», которая позволяет продолжить выполнение процесса на виртуальном процессоре, при этом синхронизируя его с реальным процессором. Таким образом, можно проводить отладку виртуального процессора, имея возможность отслеживать его работу в реальном времени.

Важно учитывать, что для корректной синхронизации необходимо иметь правильную конфигурацию *OpenOCD* для работы с конкретным виртуальным процессором и отладочным интерфейсом реального процессора. Также необходимо учитывать особенности работы с отдельными типами процессоров и отладочных интерфейсов.

Для сборки проекта для тестирования на наличие аппаратных уязвимостей возможно использовать кроссплатформенную систему автоматической сборки *CMake*, которая позволяет создавать, тестировать и пакетировать систему сборки исходных кодов, а также свободно распространяемых компиляторов[28].

Фрагмент кода сборки тестовой программы с помощью *CMake* для процессора RISC-V приведен на рис. 6. Для выполнения процедуры необходимо установить *CMake* и компилятор RISC-V. Для этого можно использовать пакетный менеджер операционной системы или загрузить их с официальных сайтов.

Для запуска необходимо в терминале выполнить команды «mkdir build && cd build && cmake .. && make», после чего выполнится сборка проекта, а результат сборки загрузится в стенд и виртуальную

[22] Architecture and Core Commands. https://openocd.org/doc/html/Architecture-and-Core-Commands.html#RISC_002dV-Authentication-Commands. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[23] Verilog-Mode Help. https://veripool.org/verilog-mode/help/. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[24] Shepherd C., Markantonakis K. *Vulnerabilities analysis and attack scenarios description.* 2021. https://exfiles.eu/wp-content/uploads/2022/07/EXFILES-D5.1-Vulnerabilities-analysis-and-attack-scenarios-description-PU-M06.pdf. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[25] Wang B. *Improving and Securing Machine Learning Systems*: Ph.D. Thesis (Comput.). USA: UC Santa Barbara; 2019. P. 10–14. https://escholarship.org/uc/item/1nv8m9nb. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[26] Guo Z. *Security of Internet of Things Devices and Networks*: Ph.D. Thesis (Comput.). USA: UC Irvine; 2016. P. 1–30. https://escholarship.org/uc/item/4rq8s4jx. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[27] Spear Ch. *System Verilog for Verification: A Guide to Learning the Testbench Language Features*. Springer. 2018. https://3ec1218usm.files.wordpress.com/2016/12/book_systemverilog_for_verification.pdf. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[28] Платформа Qualys [Qualys platform]. https://www.qualys.com/solutions/pci-compliance/. Дата обращения 15.05.2023. / Accessed May 15, 2023.

```
    delay_1ms(100);
    printf("linpackc test \r\n");
    rtc_print_current_time();
    printf("Start >> linpackc test \r\n");

    rtc_print_current_time();
    printf("Stop >> linpackc test \r\n");

    while(1) {
        time_start = rtc_get_subsecond();
        linpackc_test();
        uint16_t real_len = read_string(test_string, TEST_STR_SIZE);
        printf("len read data = %d \r\n",real_len);

        if(real_len){
            printf("read data = %s \r\n",test_string);
            memset(test_string, 0, TEST_STR_SIZE);
        }

        time_stop = rtc_get_subsecond();
        uint64_t runtime = (uint64_t)time_stop-(uint64_t)time_start;
        printf("runtime = %ld \r\n",(uint32_t)runtime);
        rtc_print_current_time();
    }
}
```

Рис. 7. Код запуска стресс-теста для проверки поведенческого описания RISC-V

систему тестирования, код описания которой приведен на рис. 7.

Поскольку рассматривается аппаратная уязвимость в вычислительном блоке FPU, одним из вариантов выявления уязвимостей является непосредственное стресс-тестирование систем, связанных с вычислениями с плавающей точкой.

Выявление аппаратных уязвимостей через стресс-тестирование может быть достаточно сложным, непроизводительным и зависит от типа уязвимости, которую пытаются обнаружить. Однако можно выделить следующие подходы:

1. Изменение условий эксплуатации: можно использовать стресс-тестирование для выявления уязвимостей, связанных с длительной работой устройства в условиях высокой нагрузки. Например, можно увеличить количество запросов к устройству, увеличить длительность работы, изменить температуру, влажность или другие параметры эксплуатации.
2. Моделирование атак: с помощью стресс-тестирования можно моделировать атаки на устройство.
3. Проверка стойкости к перегрузкам: можно использовать стресс-тестирование для проверки стойкости устройства к перегрузкам, например, проверить, как устройство справляется с высоким трафиком или ситуацией, когда число пользователей в сети резко увеличивается.
4. Использование тестовых случайных данных: с помощью генерации тестовых случайных данных можно проверить устойчивость устройства к ошибкам в данных, например, к ошибкам в передаче данных или к ошибкам в хранении данных.

Важно понимать, что стресс-тестирование может помочь выявить только определенные аппаратные уязвимости. Для полной проверки необходимо использовать комбинацию различных методов и тестов, а также следовать рекомендациям по безопасности производителя устройства. Развитием этой концепции является выявление уязвимостей посредством сканеров ПО.

Результатом проведенных исследований можно считать создание стенда для выявления аппаратных уязвимостей, а также технологии работы с ним, поскольку в доступной научной литературе нет четкого определения и описания шагов по созданию аналогичных аппаратных средств и разработке примеров для отработки возможных уязвимых ситуаций.

ВЫЯВЛЕНИЕ УЯЗВИМОСТЕЙ ПОСРЕДСТВОМ СКАНЕРОВ ПО

Сравнение наиболее часто применяющихся систем сканирования ПО на наличие уязвимостей приведено в таблице.

Одним из инструментов, который специализируется на поиске ошибок и уязвимостей в коде на Си, C++, C# и Java, является статический анализатор кода *PVS-Studio*, однако для более полного сканирования кода на языке Си могут потребоваться другие инструменты, такие как *Gitleaks*, *Trivy*[29], *Burp Suite*[30] и *MobSF*[31]. Приведем краткое описание этих инструментов и их возможностей для обнаружения уязвимостей в коде на языке Си.

[29] https://trivy.dev/. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[30] https://portswigger.net/burp. Дата обращения 15.05.2023. / Accessed May 15, 2023.

[31] https://github.com/MobSF/Mobile-Security-Framework-MobSF. Дата обращения 15.05.2023. / Accessed May 15, 2023.

Таблица. Сравнение различных систем сканирования ПО на наличие уязвимостей

№	Наименование сканера	Открытый код	Поддержка Си/С++	Основное предназначение
1	*Sn1per*[32]	нет	нет	Используется для сканирования уязвимостей в веб-приложениях, портах и сетевых устройствах. Поддерживает сканирование на основе перебора словаря и позволяет настраивать различные параметры сканирования
2	*Wapiti3*[33]	да	нет	Сканер уязвимостей для веб-приложений, который имеет открытый исходный код и может выполнять автоматический поиск уязвимостей в различных частях веб-приложения, таких как параметры URL, формы, заголовки и скрипты
3	*Nikto*[34]	да	нет	Осуществляет поиск уязвимостей на основе баз данных и расширяемых скриптов. Позволяет сканировать различные уязвимости, такие как XSS[35], SQL-инъекции, подделку заголовков, несанкционированный доступ и др.
4	*OWASP ZAP*[36]	да	нет	Предоставляет возможность сканирования автоматически или вручную, а также проведения тестов на проникновение. Используется для поиска уязвимостей, таких как SQL-инъекции, XSS, CSRF[37], некорректная авторизация и др.
5	*Sqlmap*[38]	да	нет	Инструмент для автоматического сканирования уязвимостей SQL-инъекций в веб-приложениях. Поддерживает множество баз данных, включая *MySQL*, *Oracle*, *PostgreSQL*, *Microsoft SQL Server* и др.
6	*Acunetix WVS*[39]	нет	нет	Инструмент для сканирования уязвимостей в веб-приложениях, который предоставляет возможность сканирования автоматически или вручную. Поддерживает обнаружение уязвимостей, таких как XSS, SQL-инъекции, утечки информации, нарушение безопасности файлов и др.
7	*Vega*[40]	да	нет	Инструмент для сканирования уязвимостей в веб-приложениях. Поддерживает сканирование на основе перебора словаря и предоставляет возможность проведения тестов на проникновение. Позволяет обнаруживать уязвимости, такие как XSS, SQL-инъекции
8	*PVS-Studio*[41]	нет	да	Статический анализатор кода на Си, C++, C# и Java, предназначенный для поиска ошибок и уязвимостей
9	*Gitleaks*[42]	нет	да	Инструмент для сканирования открытых репозиториев Git на наличие конфиденциальной информации и других уязвимостей безопасности. Работает путем анализа исходного кода, коммитов и истории изменений в репозитории на предмет наличия строк кода, содержащих конфиденциальную информацию, такую как пароли, ключи SSH[43], токены доступа, секреты API
10	*QARK*[44]	да	нет	Сканер приложений на Java для Android и iOS

[32] https://github.com/1N3/Sn1per/releases. Дата обращения 15.05.2023. / Accessed May 15, 2023.
[33] https://pypi.org/project/wapiti3/ (in Russ.). Дата обращения 15.05.2023. / Accessed May 15, 2023.
[34] https://github.com/sullo/nikto. Дата обращения 15.05.2023. / Accessed May 15, 2023.
[35] XSS, cross-site scripting – межсайтовый скриптинг.
[36] https://www.zaproxy.org/docs/. Дата обращения 15.05.2023. / Accessed May 15, 2023.
[37] CSRF, cross-site request forgery – межсайтовая подделка запроса.
[38] https://sqlmap.org/. Дата обращения 15.05.2023. / Accessed May 15, 2023.
[39] https://allsoft.ru/software/vendors/acunetix/acunetix-web-vulnerability-scanner-/ (in Russ.). Дата обращения 15.05.2023. / Accessed May 15, 2023.
[40] https://subgraph.com/vega/. Дата обращения 15.05.2023. / Accessed May 15, 2023.
[41] https://pvs-studio.ru/ru/pvs-studio/ (in Russ.). Дата обращения 15.05.2023. / Accessed May 15, 2023.
[42] https://github.com/gitleaks/gitleaks. Дата обращения 15.05.2023. / Accessed May 15, 2023.
[43] SSH, secure shell – безопасная оболочка.
[44] https://github.com/linkedin/qark. Дата обращения 15.05.2023. / Accessed May 15, 2023.

1. *Gitleaks* – инструмент для поиска конфиденциальных данных в Git-репозиториях, который может быть использован для сканирования кода на языке Си, хранящегося в Git.
2. *Trivy* – инструмент для сканирования контейнеров и образов Docker на наличие уязвимостей в используемых пакетах и зависимостях. Может использоваться для сканирования Docker-образов, содержащих код на языке Си.
3. *Burp Suite* – популярный инструмент для тестирования безопасности веб-приложений, который может использоваться для сканирования веб-приложений, написанных на языке Си.
4. *MobSF* – инструмент для сканирования мобильных приложений на наличие уязвимостей, который может использоваться для сканирования мобильных приложений, написанных на языке Си (например, с помощью Native Development Kit).

Каждый из этих инструментов предназначен для обнаружения уязвимостей в различных областях, и в сочетании они могут обеспечить более полное покрытие безопасности кода.

Однако следует отметить, что большинство сканеров из списка могут использоваться для сканирования кода на различных языках программирования, включая Си, но в зависимости от конкретных потребностей и типа уязвимостей, которые необходимо обнаружить, возможно, потребуется использовать сочетание нескольких инструментов.

ЗАКЛЮЧЕНИЕ

Проведенные исследования показали, что предложенный подход на основе систем сканирования и полунатурного моделирования успешно выявляет аппаратные уязвимости цифровых систем в тех случаях, когда другие методы оказались неэффективными, а анализ результатов трудно интерпретируется. Только в синтетическом эксперименте с помощью полунатурного моделирования удалось сузить область поиска и идентифицировать систему со встроенным зловредным кодом с уязвимостью. Результаты экспериментов позволили выработать методологию и определить набор инструментов для выявления уязвимых мест цифровых устройств вычислительных систем, а также создать библиотеку готовых решений для реализации оптимального решения.

Полученные результаты и разработанный стенд для проведения экспериментов могут использоваться в перспективных проектах по созданию цифровых устройств на современной элементной базе с возможностью перехода на новые технологии в части выявления аппаратных уязвимостей, т.к. безопасность аппаратного обеспечения необходимо рассматривать как приоритетную задачу в различных отраслях и сферах деятельности, а обнаружение и устранение уязвимостей цифровых компонентов устройств должны проводиться как на ранних этапах разработки, так и на этапе эксплуатации систем.

БЛАГОДАРНОСТИ

Работа выполнена при поддержке Министерства науки и высшего образования РФ (Государственное задание для университетов № ФГФЗ-2023-0005) и с применением оборудования Центра коллективного пользования РТУ МИРЭА (соглашение от 01.09.2021 № 075-15-2021-689, уникальный идентификационный номер 2296.61321X0010).

ACKNOWLEDGMENTS

This work was supported by the Ministry of Science and Higher Education of the Russian Federation (State task for universities No. FGFZ-2023-0005) and using the equipment of the Center for Collective Use of RTU MIREA (agreement dated September 01, 2021, No. 075-15-2021-689, unique identification number 2296.61321X0010).

Вклад авторов

Е.Ф. Певцов – идея исследования, консультации по вопросам исследования, написание текста статьи.

Т.А. Деменкова – идея исследования, планирование исследования, научное редактирование статьи.

А.О. Индришенок – идея исследования, проведение исследования, написание текста статьи, интерпретация и обобщение результатов.

В.В. Филимонов – консультации по вопросам исследования, написание текста статьи.

Authors' contributions

E.F. Pevtsov – the research idea, consultations on research issues, writing the text of the article.

T.A. Demenkova – the research idea, research planning, scientific editing of the article.

A.O. Indrishenok – the research idea, conducting research, writing the text of the article, interpretation and generalization of the results.

V.V. Filimonov – consultations on research issues, writing the text of the article.

СПИСОК ЛИТЕРАТУРЫ

1. Smetana D. FPGA-Enabled Trusted Boot Is Part of Building Security into Every Aspect of Trusted Computing Architectures. *Military & Aerospace Electronics Journal*. September 25, 2019. URL: https://www.militaryaerospace.com/trusted-computing/article/14040672/trustedcomputing-embedded-computing-realworld
2. Сесин И.Ю., Болбаков Р.Г. Сравнительный анализ методов оптимизации программного обеспечения для борьбы с предикацией ветвлений на графических процессорах. *Russ. Technol. J*. 2021;9(6):7–15. https://doi.org/10.32362/2500-316X-2021-9-6-7-15
3. Shayan M., Basu K., Karri R. Hardware Trojans Inspired Hardware IP Watermarks. *IEEE Design & Test*. 2019;36(6):72–79. https://doi.org/10.1109/MDAT.2019.2929116
4. Hennessy J.L., Patterson D.A. A new golden age for computer architecture: Domain-specific hardware/software co-design, enhanced security, open instruction sets, and agile chip development. In: *Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture* (*ISCA*). IEEE; 2018. https://doi.org/10.1109/ISCA.2018.00011
5. Li D., Zhang Q., Zhao D., Li L., He J., Yuan Y., Zhao Y. Hardware Trojan Detection Using Effective Property-Checking Method. *Electronics*. 2022;11(17):2649. https://doi.org/10.3390/electronics11172649
6. Алехин В.А. Проектирование электронных систем с использованием SystemC и SystemC–AMS. *Russ. Technol. J*. 2020;8(4):79–95. https://doi.org/10.32362/2500-316X-2020-8-4-79-95
7. Yang K., Zhang K., Ren J., Shen X. Security and privacy in mobile crowdsourcing: Challenges and opportunities. *IEEE Commun. Mag*. 2015;53(8):75–81. https://doi.org/10.1109/MCOM.2015.7180511
8. Lou X., Zhang T., Jiang J., Zhang Y. *A Survey of Microarchitectural Side-channel Vulnerabilities, Attacks and Defenses in Cryptography*. Vol. 1. No. 1. March 2021. URL: https://arxiv.org/pdf/2103.14244
9. Skorobogatov S., Woods C. Breakthrough Silicon Scanning Discovers Backdoor in Military Chip. In: Prouff E., Schaumont P. (Eds.). *Cryptographic Hardware and Embedded Systems – CHES 2012*. *Lecture Notes in Computer Science*. 2012. V. 7428. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-33027-8_2
10. Tasiran S., Keutzer K. Coverage metrics for functional validation of hardware designs. *IEEE Des. Test. Comput.* 2001;18(4):36–45. https://doi.org/10.1109/54.936247
11. Mukhopadhyay D., Chakraborty R.S. *Hardware Security: Design, Threats, and Safeguards.* CRC Press; 2014. 542 p. ISBN 978-1-4398-9584-9
12. Тарасов И.Е. *ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования.* М.: Горячая линия – Телеком; 2024. 538 с. ISBN 978-5-9912-0802-4
13. Turkington K., Masseios K., Constantinides G.A., Leong P. FPGA Based Acceleration of the Linpack Benchmark: A High Level Code Transformation Approach. In: *2006 International Conference on Field Programmable Logic and Applications*. IEEE; 2007. https://doi.org/10.1109/FPL.2006.311240
14. Tamuly S., Joseph A. Chandrasekharan J. Deep Learning Model for Image Classification. In: Smys S., Tavares J., Balas V., Iliyasu A. (Eds.). *Computational Vision and Bio-Inspired Computing. ICCVBIC 2019. Advances in Intelligent Systems and Computing*. Springer, Cham; 2019. V. 1108. P. 312–320. https://doi.org/10.1007/978-3-030-37218-7_36
15. Majeric F., Gonzalvo B., Bossuet L. JTAG Fault Injection Attack. *IEEE Embed. Syst. Lett.* 2018;10(3):65–68. https://doi.org/10.1109/LES.2017.2771206
16. Abdalhag B., Awad A., Hawash A. A fast Binary Decision Diagram (BDD)-based reversible logic optimization engine driven by recent meta-heuristic reordering algorithms. *Microelectron. Reliab*. 2021;123:114168. https://doi.org/10.1016/j.microrel.2021.114168
17. Певцов Е.Ф., Деменкова Т.А., Шнякин А.А. Тестопригодное проектирование интегральных схем и проблемы защиты проектов. *Russ. Technol. J*. 2019;7(4):60–70. https://doi.org/10.32362/2500-316X-2019-7-4-60-70
18. Kuo M.-H., Hu Ch.-M., Lee K.-J. Time-Related Hardware Trojan Attacks on Processor Cores. In: *IEEE International Test Conference in Asia* (*ITC-Asia*). IEEE; 2019. https://doi.org/10.1109/ITC-Asia.2019.00021
19. Комолов Д., Золотухо Р. Использование микросхем специальной памяти для обеспечения защиты ПЛИС FPGA от копирования. *Компоненты и технологии*. 2008;12:24–26. URL: https://kit-e.ru/wp-content/uploads/2008_12_24.pdf
20. Becker A., Hu W., Tai Y., Brisk P., Kastner R., Ienne P. Arbitrary Precision and Complexity Tradeoffs for Gate-Level Information Flow Tracking. In: *Proceedings of the 54th ACM/EDAC/IEEE Design Automation Conference* (*DAC*). IEEE, 2017. Part 128280. https://doi.org/10.1145/3061639.3062203
21. Polychronou N.F., Thevenon P.H., Puys M., Beroulle V. A Comprehensive Survey of Attacks without Physical Access Targeting Hardware Vulnerabilities in IoT/IIoT Devices, and Their Detection Mechanisms. *ACM Trans. Design Automat. Electron. Syst*. 2022;27(1):1–35. https://doi.org/10.1145/3471936
22. Erata F., Deng Sh., Zaghloul F., Xiong W., Demir O., Szefer J. Survey of Approaches and Techniques for Security Verification of Computer Systems. *ACM J. Emerg. Technol. Comput. Syst.* 2022;1(1):Article 1. https://doi.org/10.1145/3564785
23. Yang X., Zhao D., Jiang Y., Zhang X., Yuan Y. Fault Simulation and Formal Analysis in Functional Safety CPU FMEDA Campaign. *J. Phys.: Conf. Ser*. 2021;1769:012061. https://doi.org/10.1088/1742-6596/1769/1/012061

REFERENCES

1. Smetana D. FPGA-Enabled Trusted Boot Is Part of Building Security into Every Aspect of Trusted Computing Architectures. *Military & Aerospace Electronics Journal*. September 25, 2019. Available from URL: https://www.militaryaerospace.com/trusted-computing/article/14040672/trustedcomputing-embedded-computing-realworld
2. Sesin I.Yu., Bolbakov R.G. Comparative analysis of software optimization methods in context of branch predication on GPUs. *Russ. Technol. J.* 2021;9(6):7–15 (in Russ.). https://doi.org/10.32362/2500-316X-2021-9-6-7-15
3. Shayan M., Basu K., Karri R. Hardware Trojans Inspired Hardware IP Watermarks. *IEEE Design & Test*. 2019;36(6):72–79. https://doi.org/10.1109/MDAT.2019.2929116
4. Hennessy J.L., Patterson D.A. A new golden age for computer architecture: Domain-specific hardware/software co-design, enhanced security, open instruction sets, and agile chip development. In: *Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture* (*ISCA*). IEEE; 2018. https://doi.org/10.1109/ISCA.2018.00011
5. Li D., Zhang Q., Zhao D., Li L., He J., Yuan Y., Zhao Y. Hardware Trojan Detection Using Effective Property-Checking Method. *Electronics*. 2022;11(17):2649. https://doi.org/10.3390/electronics11172649
6. Alekhin V.A. Designing electronic systems using SystemC and SystemC-AMS. *Russ. Technol. J.* 2020;8(4):79–95 (in Russ.). https://doi.org/10.32362/2500-316X-2020-8-4-79-95
7. Yang K., Zhang K., Ren J., Shen X. Security and privacy in mobile crowdsourcing: Challenges and opportunities. *IEEE Commun. Mag*. 2015;53(8):75–81. https://doi.org/10.1109/MCOM.2015.7180511
8. Lou X., Zhang T., Jiang J., Zhang Y. *A Survey of Microarchitectural Side-channel Vulnerabilities, Attacks and Defenses in Cryptography*. Vol. 1. No. 1. March 2021. Available from URL: https://arxiv.org/pdf/2103.14244
9. Skorobogatov S., Woods C. Breakthrough Silicon Scanning Discovers Backdoor in Military Chip. In: Prouff E., Schaumont P. (Eds.). *Cryptographic Hardware and Embedded Systems – CHES 2012*. *Lecture Notes in Computer Science*. 2012. V. 7428. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-33027-8_2
10. Tasiran S., Keutzer K. Coverage metrics for functional validation of hardware designs. *IEEE Des. Test. Comput.* 2001;18(4):36–45. https://doi.org/10.1109/54.936247
11. Mukhopadhyay D., Chakraborty R.S. *Hardware Security: Design, Threats, and Safeguards.* CRC Press; 2014. 542 p. ISBN 978-1-4398-9584-9
12. Tarasov I.E. *PLIS Xilinx. Yazyki opisaniya apparatury VHDL i Verilog, SAPR, priemy proektirovaniya* (*FPGA Xilinx. Hardware Description Languages VHDL and Verilog, CAD, Design Techniques*). Moscow: Goryachaya liniya – Telekom; 2024. 538 p. (in Russ.). ISBN 978-5-9912-0802-4
13. Turkington K., Masseios K., Constantinides G.A., Leong P. FPGA Based Acceleration of the Linpack Benchmark: A High Level Code Transformation Approach. In: *2006 International Conference on Field Programmable Logic and Applications*. IEEE; 2007. INSPEC Accession Number: 9604301. https://doi.org/10.1109/FPL.2006.311240
14. Tamuly S., Joseph A. Chandrasekharan J. Deep Learning Model for Image Classification. In: Smys S., Tavares J., Balas V., Iliyasu A. (Eds.). *Computational Vision and Bio-Inspired Computing. ICCVBIC 2019. Advances in Intelligent Systems and Computing*. Springer, Cham; 2019. V. 1108. P. 312–320. https://doi.org/10.1007/978-3-030-37218-7_36
15. Majeric F., Gonzalvo B., Bossuet L. JTAG Fault Injection Attack. *IEEE Embed. Syst. Lett.* 2018;10(3):65–68. https://doi.org/10.1109/LES.2017.2771206
16. Abdalhag B., Awad A., Hawash A. A fast Binary Decision Diagram (BDD)-based reversible logic optimization engine driven by recent meta-heuristic reordering algorithms. *Microelectron. Reliab*. 2021;123:114168. https://doi.org/10.1016/j.microrel.2021.114168
17. Pevtsov E.F., Demenkova T.A., Shnyakin A.A. Design for Testability of Integrated Circuits and Project Protection Difficulties. *Russ. Technol. J*. 2019;7(4):60–70 (in Russ.). https://doi.org/10.32362/2500-316X-2019-7-4-60-70
18. Kuo M.-H., Hu Ch.-M., Lee K.-J. Time-Related Hardware Trojan Attacks on Processor Cores. In: *IEEE International Test Conference in Asia* (*ITC-Asia*). IEEE; 2019. https://doi.org/10.1109/ITC-Asia.2019.00021
19. Komolov D., Zolotukho R. Using special memory chips to ensure FPGA copy protection. *Komponenty i tekhnologii = Components & Technologies*. 2008;12:24–26 (in Russ.). Available from URL: https://kit-e.ru/wp-content/uploads/2008_12_24.pdf.
20. Becker A., Hu W., Tai Y., Brisk P., Kastner R., Ienne P. Arbitrary Precision and Complexity Tradeoffs for Gate-Level Information Flow Tracking. In: *Proceedings of the 54th ACM/EDAC/IEEE Design Automation Conference* (*DAC*). IEEE, 2017. Part 128280. https://doi.org/10.1145/3061639.3062203
21. Polychronou N.F., Thevenon P.H., Puys M., Beroulle V. A Comprehensive Survey of Attacks without Physical Access Targeting Hardware Vulnerabilities in IoT/IIoT Devices, and Their Detection Mechanisms. *ACM Trans. Design Automat. Electron. Syst*. 2022;27(1):1–35. https://doi.org/10.1145/3471936
22. Erata F., Deng Sh., Zaghloul F., Xiong W., Demir O., Szefer J. Survey of Approaches and Techniques for Security Verification of Computer Systems. *ACM J. Emerg. Technol. Comput. Syst.* 2022;1(1):Article 1. https://doi.org/10.1145/3564785
23. Yang X., Zhao D., Jiang Y., Zhang X., Yuan Y. Fault Simulation and Formal Analysis in Functional Safety CPU FMEDA Campaign. *J. Phys.: Conf. Ser*. 2021;1769:012061. https://doi.org/10.1088/1742-6596/1769/1/012061

Об авторах

Певцов Евгений Филиппович, к.т.н., директор структурного подразделения «Центр проектирования интегральных схем, устройств наноэлектроники и микросистем», ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: pevtsov@mirea.ru. Scopus Author ID 6602652601. ResearcherID M-2709-2016, SPIN-код РИНЦ 1410-2483, http://orcid.org/0000-0001-6264-1231

Деменкова Татьяна Александровна, к.т.н., доцент, кафедра вычислительной техники, Институт информационных технологий, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: demenkova@mirea.ru. Scopus Author ID 57192958412, ResearcherID AAB-3937-2020, SPIN-код РИНЦ 3424-7489, http://orcid.org/0000-0003-3519-6683

Индришенок Александр Олегович, аспирант, кафедра вычислительной техники, Институт информационных технологий, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: indrishenoksasha@mail.ru. SPIN-код РИНЦ 2308-7140, http://orcid.org/0000-0003-1471-9043

Филимонов Владимир Викторович, старший преподаватель, кафедра физики и технической механики, Институт перспективных технологий и индустриального программирования, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: filimonov@mirea.ru. Scopus Author ID 7102525379, http://orcid.org/0000-0003-1118-6608

About the authors

Evgeniy F. Pevtsov, Cand. Sci. (Eng.), Director of Center for the Design of Integrated Circuits, Nanoelectronics Devices and Microsystems, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: pevtsov@mirea.ru. Scopus Author ID 6602652601. ResearcherID M-2709-2016, RSCI SPIN-code 1410-2483, http://orcid.org/0000-0001-6264-1231

Tatyana A. Demenkova, Cand. Sci. (Eng.), Associated Professor, Computer Technology Department, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: demenkova@mirea.ru. Scopus Author ID 57192958412, ResearcherID AAB-3937-2020, RSCI SPIN-code 3424-7489, http://orcid.org/0000-0003-3519-6683

Alexander O. Indrishenok, Postgraduate Student, Computer Technology Department, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: indrishenoksasha@mail.ru. RSCI SPIN-code 2308-7140, http://orcid.org/0000-0003-1471-9043

Vladimir V. Filimonov, Senior Lecturer, Department of Physics and Technical Mechanics, Institute for Advanced Technologies and Industrial Programming, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: filimonov@mirea.ru. Scopus Author ID 7102525379. http://orcid.org/0000-0003-1118-6608