ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
СПбГУТ))) Сентябрь 2016 Том 4 № 3 ISSN 2307-1303 http://www.itt.sut.ru
TELECOM IT
September 2016 Vol. 4 Iss. 3 ISSN 2307-1303 http://www.itt.sut.ru

КАТЕГОРИАЛЬНЫЙ СИНТЕЗ И ТЕХНОЛОГИЧЕСКИЙ АНАЛИЗ ВАРИАНТОВ БЕЗОПАСНОГО ИМПОРТОЗАМЕЩЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ТЕЛЕКОММУНИКАЦИОННЫХ УСТРОЙСТВ

М. В. Буйневич[1*], К. Е. Израилов[1]

[1] СПбГУТ, Санкт-Петербург, 193232, Российская Федерация
* Адрес для переписки: bmv1958@yandex.ru

Аннотация
Предмет исследования. Статья посвящена подходам к безопасному импортозамещению программного обеспечения телекоммуникационных устройств. **Метод**. Для получения научно-обоснованного пула вариантов импортозамещения используется категориальный синтез в его классификационном смысле. Также применятся технологический анализ, под которым здесь понимается качественная оценка синтезированных вариантов по критериям результативности, детерминированности и массовости. **Основные результаты.** С учетом 2-х категориальных пар синтезировано 4 основных способа импортозамещения и 4 основных способа поиска уязвимостей в программном обеспечении. Их взаимное наложение позволяет получить 16 потенциально различных вариантов осуществления безопасного импортозамещения. **Практическая значимость.** Полный пул технологически оцененных вариантов позволяет повысить обоснованность принятия решений в области научно-технической политики.
Ключевые слова
программное обеспечение, телекоммуникационные устройства, импортозамещение, безопасность, варианты, категориальный синтез, технологический анализ.

Информация о статье
УДК 004.42:621.39
Язык статьи – русский.
Поступила в редакцию 07.07.2016, принята к печати 26.08.16.

Ссылка для цитирования: Буйневич М. В., Израилов К. Е. Категориальный синтез и технологический анализ вариантов безопасного импортозамещения программного обеспечения телекоммуникационных устройств // Информационные технологии и телекоммуникации. 2016. Том 4. № 3. С. 95–106.

CATEGORICAL SYNTHESIS AND TECHNOLOGICAL ANALYSIS OF SECURITY IMPORT SUBSTITUTION VARIANTS OF TELECOMMUNICATION DEVICES SOFTWARE

M. Buinevich[1*], K. Izrailov[1]

[1] SPbSUT, St. Petersburg, 193232, Russian Federation
* Corresponding author: bmv1958@yandex.ru

Abstract—Research subject. The article is dedicated to import substitution approach to security software telecommunications devices. **Method.** It is used the categorical synthesis in its classification sense for science-based import options pool. Also it is applied the technological analysis which meaning is a qualitative assessment of the synthesized variants in scoring criteria, determinism and mass. **Core results.** It was synthesized 4 main ways to import and 4 main ways to find vulnerabilities in software subject to the 2-categorical pairs. Their overlap allows you to get potentially 16 different variants of a realization of secure import substitution. **Practical relevance.** Full pool of technologically priced variants allows improving the validity of the adoption in the field of science and technology policy-making.
Keywords—Software, telecommunication devices, import substitution, security, variants, categorical synthesis, technological analysis.

Article info
Article in Russian.
Received 07.07.16, accepted 26.08.16.

For citation: Buinevich M., Izrailov K.: Categorical synthesis and technological analysis of security import substitution variants of telecommunication devices software // Telecom IT. 2016. Vol. 4. Iss. 3. pp. 95–106 (in Russian).

Введение

В свете последних тенденций в экономике и политике вопросы импортозамещения программно-аппаратных средств современных инфотелекоммуникационных технологий стоят особенно остро [1, 2, 3]. Импортозамещение, применительно к области программного обеспечения (ПО), означает создание собственных программных продуктов внутри страны. Особенностями области телекоммуникационного оборудования (ТКУ), вносящими определенные коррективы в процесс импортозамещения ПО, являются следующие. Во-первых, исходный код, как правило, отсутствует. Во-вторых, ТКУ предоставляет функционал, плохо тестируемый извне – отсутствуют графические интерфейсы (только консольная строка и конфигурационные файлы), а входные/выходные данные имеют высокую вариативность по размеру и содержанию (сетевые протоколы). И, в-третьих, уязвимости в ПО ТКУ напрямую влияют на безопасность обраба-

тываемой им информации, нарушение конфиденциальности, целостности и доступности которой может приводить к критическим последствиям [4, 5, 6].

Ввиду многообразия и сложности процесса создания ПО ТКУ подходы к его импортозамещению, скорее всего, вариативны. И для принятия решений о предпочтении желательно иметь полный, научно-обоснованный пул вариантов. Для его формирования можно воспользоваться методом категориального синтеза, который в контексте поставленной задачи предполагает вычленение категорий, наиболее существенных для настоящей предметной области, с генерацией на их основе неких достаточно абстрактных способов достижения цели и последующей их трактовкой в терминах, как минимум, двух взаимопересекающихся предметных областей – создания собственного (отечественного) ПО и поиска в нем уязвимостей. Их комбинация призвана синтезировать все возможные варианты создания отечественного ПО без уязвимостей или варианты, так называемого, безопасного импортозамещения.

Принятие решений о предпочтении будет, несомненно, опираться на оценку технологичности того или иного варианта, так как сфера создания ПО ТКУ является сверхтехнологичной. Такая предварительная качественная оценка может быть получена на путях анализа синтезированных вариантов на предмет их соответствия неким критериям технологичности. К таким критериям следует отнести результативность, детерминированность и массовость (в предположении, что их содержание вытекает из применяемого термина, оно здесь опущено). Оценка же эффективности, являясь доминирующей при любом выборе, не относится напрямую к оценке технологичности и определяется как результативность соотнесенная с затратами; причем здесь для изменения результативности нередко применяются нравственные, юридические и иные критерии выбора, который лежит в плоскости научно-технической политики, проводимой государством и выходят за рамки статьи.

Категориальный синтез вариантов

Определим возможные вариации процесса импортозамещения ПО ТКУ с помощью рассмотренных категориальных пар. Разработка собственного ПО может быть осуществлена следующим образом: во-первых, ее можно произвести полностью «с нуля», а, во-вторых, можно использовать существующие (в том числе, импортные) наработки. С другой стороны, в качестве алгоритмической базы для разработки могут быть взяты, как аналоги в виде открытого ПО, так и существующие закрытые (проприетарные) продукты. С учетом категориальных пар: «разработка *vs* доработка (модификация)» и «открытое *vs* закрытое (проприетарное)», – обоснованным будет деление возможностей импортозамещения ПО ТКУ на 4 способа (далее – способ IS-N, где IS от Import Substitution, N – условный номер), каждый из которых имеет собственный подход к реализации.

Способ IS-1. Создание «с нуля» на базе открытого ПО соответствует типичной разработке нового программного продукта на основании открытой информации об алгоритмах замещаемого продукта, примерах его кода, программной документации и т. п.

Способ IS-2. Доработка существующего продукта на базе открытого ПО означает взятие за основу открытого исходного кода и его модификацию в соответствии с функционалом замещаемого продукта.

Способ IS-3. Создание «с нуля» на базе закрытого ПО требует проведение полного реверс-инжиниринга (т. е. обратной разработки) замещаемого продукта, получение его архитектуры и алгоритмов с последующим кодирование собственного.

Способ IS-4. Доработка существующего продукта на базе закрытого ПО осуществляется путем частичного реверс-инжиниринга ключевых элементов замещаемого ПО с их повторным кодированием, а также реализация собственных частей продукта, реализующих недостающий функционал. Такими ключевыми элементами могут быть, например, криптомодули или иные алгоритмы обеспечения безопасности информации.

Возможные способы импортозамещения ПО ТКУ представлены в табл. 1.

Таблица 1.

Способы импортозамещения ПО ТКУ

		Способ разработки	
		Создание с нуля	*Доработка*
База разработки	*Открытое ПО*	1. Проектирование и кодирование нового продукта	2. Модификация открытого исходного кода
	Закрытое ПО	3. Полный реверс-инжиниринг с перекодированием	4. Частичный реверс-инжиниринг с реализацией недостающего функционала

У каждого способа есть свои сильные и слабые стороны.

Способ IS-1 позволяет получить новый продукт без заимствований идей сторонних реализаций; однако, такой путь имеет огромные трудозатраты, поскольку требует развития целых направлений, отстающих от импортных аналогов. Например, разработка продуктов ТКУ для распределенных БД с применением собственного языка запросов (аналога SQL) потребует как развитие этого самого языка, так и общей идеологии работы с информационными хранилищами.

Способ IS-2 значительно снизит трудозатраты способа IS-1, поскольку будет использовать открытый код дорабатываемых продуктов. Тем не менее, встает вопрос, как лицензионных соглашений использования кода, так и возможных уязвимостей на стыке исходного и нового кода. Также немаловажным является и то, что далеко не весь требуемый функционал может иметь открытые аналоги. В частности, для большинства ПО ТКУ не существует открытого ПО из-за коммерческих соображений поставщиков.

Способ IS-3 изначально говорит о том, что уже существует продукт, предоставляющий требуемый функционал, и необходима разработка его аналога внутри страны – т. е. требуется лишь некий механизм переработки и «обезопасивания» кода. Таким механизмом может быть применение декомпиляции (частный случай реверс-инжиниринга), позволяющее сначала получить исходный код, а затем с адекватными трудозатратами ручным способом его

проверить и модифицировать, получив, по сути, собственный продукт. Тем не менее, декомпиляция в применимой форме возможна лишь теоретически и для определенного класса кода. Так, данный способ в ТКУ применим для замещения импортных HTTP-серверов собственными, но плохо подойдет при реализации VPN-механизма по причине сложности и закрытости реализаций его криптоалгоритмов.

Способ IS-4 можно считать упрощенным вариантом предыдущего. С одной стороны он уже берет за основу существующий импортный продукт. С другой стороны в нем требуется реверс-инжиниринг лишь для ключевых частей кода. И, с третьей, недостающий код, который обязан быть собственной разработки, реализуется «с нуля». Однако очевидно, что применение в одном способе разнородных подходов усложнит общий процесс разработки и приведет в конечном итоге к появлению ошибок.

Помимо задачи создания собственного аналога продукта важной проблемой будет появление в нем уязвимостей – ошибок в ПО, носящих случайных или злонамеренный характер [7]. И если первые присутствуют практически в любом продукте и вносят лишь небольшой «шум» в корректность общей работы, то вторые могут нанести существенный урон именно по причине их появления – действий злоумышленников.

Существует ограниченное количество распространенных способов поиска уязвимостей в ПО, состоящих из процесса применения и объекта применения [8]. Процесс, по его временной характеристике, может быть статическим – применимым к самому содержанию кода, и динамическим – относящимся к его реальному выполнению (т. е. рассмотрение эффекта кода). Объект же применения может быть разделен по его отношению к использующему, а именно на человеко- и машинно-ориентированный. Для существующей области IT можно к первым отнести исходный код – разрабатывается человеком, и машинный код – выполняется средой. Совокупность указанных делений на категориальные пары: «статический *vs* динамический» и «человек *vs* автомат (машина)», – приводит к существованию 4-х способов (далее – способ SV-N, где SV от *Search for Vulnerabilities*, N – условный номер), поиска уязвимостей, приведенных в табл. 2.

Таблица 2.

Способы поиска уязвимостей в ПО ТКУ

		Процесс применения	
		Статический	*Динамический*
Объект применения	*Человеко-ориентированный*	1. Статический анализ исходного кода	2. Динамический анализ исходного кода (пока не существует!)
	Машинно-ориентированный	3. Статический анализ машинного кода	4. Динамический анализ машинного кода

Способ SV-1 существует в виде ручного труда экспертов и ряда автоматизирующих средств для поиска уязвимостей напрямую по исходному коду. Важным достоинством способа является теоретическая возможность определения всех уязвимостей и условная простота самого процесса – работа с формализо-

ванным текстовым представлением. Однако существенный его недостаток, помимо высокой ручной трудоемкости, состоит в отсутствии исходных кодов для большинства ТКУ.

Способ SV-2, дословно означающий выполнение исходного кода, на данный момент не существует. Это связано, как с малой изученностью теории представлений программного кода и их особенностей, так и с отсутствием сведений об успешных практических попытках ее применения.

Способ SV-3 аналогичен способу SV-1, но применяется для машинного кода. С одной стороны, такой анализ можно проводить в условиях отсутствия исходного кода. С другой стороны, объем машинного кода в разы превосходит аналогичный ему исходный, а применение ручного труда имеет чрезмерную трудоемкость.

Способ SV-4, по сути, означает выполнение машинного кода в эмуляторах среды выполнения или же ведение трассировочных логов при запуске ПО на реальном ТКУ. Способ в основном имеет ручную форму, хотя возможно использование автоматизирующих средств. Важно отметить, что обнаружение уязвимостей по их эффектам при выполнении сократит время поиска. Существенным недостатком способа является то, что не все варианты логики работы могут быть воспроизведены экспертом, а это означает, что не все уязвимости будут обнаружены. Также способ плохо подходит для обнаружения программных закладок, поскольку последние изначально встраиваются, чтобы до некоторого события не проявлять себя.

Произведем наложение описанных способов импортозамещения на существующие способы поиска уязвимостей. Согласно правил комбинаторики это позволит получить 16 потенциально различных вариантов осуществления безопасного импортозамещения ПО ТКУ (табл. 3).

Таблица 3.

Варианты осуществления безопасного импортозамещения ПО ТКУ

		Способы импортозамещения ПО			
		1. Проектирование и кодирование нового продукта	*2. Модификация открытого исходного кода*	*3. Полный реверс-инжиниринг с перекодированием*	*4. Частичный реверс-инжиниринг с реализацией недостающего функционала*
Способы поиска уязвимостей в ПО ТКУ	*1. Статический анализ исходного кода*	1.1. Отсутствует эффект	1.2. Поиск в открытом исходном коде	1.3. Поиск в восстановленном псевдо-коде	1.4. Поиск в восстановленном псевдо-коде ключевых частей
	2. Динамический анализ исходного кода	2.1. Не существует	2.2. Не существует	2.3. Не существует	2.4. Не существует
	3. Статический анализ машинного кода	3.1. Не применимо	3.2. Не применимо	3.3. Поиск в машинном коде	3.4. Поиск в машинном коде ключевых частей
	4. Динамический анализ машинного кода	4.1. Не применимо	4.2. Не применимо	4.3. Анализ выполнения машинного кода	4.4. Анализ выполнения машинного кода ключевых частей

Технологический анализ вариантов

Оценим технологичность синтезированных вариантов по введенным ранее критериям. При этом несуществующие и заведомо неприменимые варианты из рассмотрения исключаются.

Вариант 1.1 не имеет эффекта в смысле поиска уязвимостей, поскольку новый продукт разрабатывается полностью на базе собственного исходного кода, не приводя к их появлению. Будем считать, что код, созданный собственными усилиями внутри страны (и под контролем государства), не имеет критичных уязвимостей и поэтому далее из рассмотрения исключается.

Вариант 1.2 соответствует автоматизированному поиску уязвимостей в открытом исходном коде, который в дальнейшем модифицируется. Как правило, такой код многократно проверен разработчиками и оттестирован пользователями, что значительно снижает наличие в нем уязвимостей. Тем не менее, специальные требования к замещающему продукту могут приводить к дополнительным проверкам кода. В случае изначальной «популярности» открытого исходного кода и достаточной квалификации разработчиков дорабатываемого кода, результат от применения варианта будет высоким. Весь процесс создания такого «собственного» продукта хорошо разбивается на крупные этапы: поиск кода, его модификация и тестирование нового функционала, – хотя каждый в отдельности требует плохо формализуемой творческой реализации. Вариант можно считать популярным лишь при наличии открытых источников с кодом.

Вариант 1.3 требует восстановления готового продукта, как правило, в виде псевдо-кода. При этом уязвимости в рабочем состоянии продукта будут также присутствовать и в его коде, а их поиск возможен лишь ручным трудоемким способом. Перекодирование восстановленного кода считается не вносящим уязвимостей, поскольку это производится собственными средствами внутри страны (по аналогии с вариантом 1.1). Вариант обладает низкой результативность, поскольку в восстановленном коде присутствуют все уязвимости начального, которые могут быть найдены его анализом – как правило, трудоемким ручным. И хотя для восстановления всего кода возможно применение автоматизирующих средств, тем не менее, такой способ применяется редко.

Вариант 1.4 является упрощенным вариантом 1.3, поскольку производится поиск уязвимостей в псевдо-коде лишь ключевых частей продукта, полученных с помощью реверс-инжиниринга. Перекодирование восстановленного кода, как и его доработка, считаются не вносящими уязвимостей (по аналогии с вариантом 1.1). Тот факт, что восстанавливаются лишь ключевые части продукта, а остальные разрабатываются собственными силами, позволяет утверждать о снижении количества «наследованных» уязвимостей по сравнению с Вариантом 1.3, что повышает итоговую безопасность. Трудоемкость, тем не менее, повышается. Детерминированность варианта все так же не высока, т. к. процесс реализации недостающих частей мало поддается разбиению и детализации. Вариант можно отчасти считать популярным лишь для задач, когда создание «с нуля» продукта невозможно по причине полного отсутствия информации о некотором функционале замещаемого ПО – в этом случае, такая реализация такого функционала осуществляется перекодированием.

Варианты 2.1–2.4 на данный момент развития области IT не существуют, поскольку непосредственное выполнение исходного кода ТКУ не имеет реаль-

ной практики. Скриптовые языки – такие, как Ruby, Perl, Java, C# – хотя и можно считать формально выполняемыми, однако их использование в ТКУ носит единичный характер. Тем не менее, исследование данного направления поиска уязвимостей может создать новую перспективную технологию.

Варианты 3.1 и 3.2 не применимы по причине того, что разработка продукта на базе исходного кода не содержит машинного кода, что делает соответствующую технологию поиска уязвимостей не имеющей смысла.

Вариант 3.3 позволяет перед или в процессе реверс-инжиниринга машинного кода производить его статический анализ на предмет наличия уязвимостей. Обезвреживание последних позволит получить псевдо-код уже без уязвимостей, что сделает создаваемый по нему код замещающего продукта более безопасным. Тем не менее, автоматический поиск уязвимостей по машинному коду имеет низкую результативность (в процентах найденных уязвимостей), а применение ручного труда – высокие трудозатраты. Результативность способа может быть высокой лишь в случае совместного применения реверс-инжиниринга и поиска уязвимостей в машинном коде, поскольку используемые в них подходы работают эффективно лишь на едином представлении. По аналогичным соображениям вариант до момента перекодирования обладает высокой детерминированностью. Как и все способы импортозамещения с применением полного реверс-инжиниринга, данный способ практически не применяется, хотя и имеет перспективные прогнозы.

Вариант 3.4 отличается от варианта 3.3 повышением результативности поиска за счет применения статического анализа лишь для ключевых частей кода, поскольку реализация недостающих считается не вносящей уязвимостей (по аналогии с вариантом 1.1). Его детерминированность и массовость можно считать аналогичными.

Варианты 4.1 и 4.2 не применимы по аналогичным с вариантами 3.1 и 3.2 соображениям.

Вариант 4.3 означает полноценное тестирование импортного продукта в виде «черного ящика». Процесс может быть трудоемким ручным или менее результативным, но автоматическим. Тем не менее, в обоих случаях процент найденных уязвимостей будет невысок – некоторый функционал попросту не получиться задействовать без понимания деталей реализации. Также будет стоять задача позиционировании найденных уязвимостей в машинном коде перед его реверс-инжинирингом. Автоматическое тестирование достаточно часто применяется, поскольку возможно написание целых наборов тестов по собственным требованиям или открытой документации; при этом, тесты могут применяться и для вновь разработанного продукта. Ручное же тестирование, вместе с восстановленным кодом, позволит лучше понять принципы работы замещаемого ПО и, следовательно, более эффективно создать новый продукт.

Вариант 4.4 хотя и аналогичен варианту 4.3, однако носит упрощенную форму, повышающую эффективность обнаружения уязвимостей. Так, возможно проведение динамического тестирования для ключевых частей импортного продукта (например, модулей криптографии). Для этого возможно использование не предусмотренных продуктом интерфейсов, а его внутреннего API; также целесообразно внедрение собственных тестирующих агентов непосредственно в машинной код продукта (например, для «фаззинга памяти»). Детерминированность варианта аналогична предыдущему. И хотя его использование не

слишком распространено, тем не менее, в ряде областей он может считаться перспективным.

Заключение

Применение категориального деления для сложных систем позволило выделить все возможные способы осуществления импортозамещения и поиска уязвимостей для ПО ТКУ. Попарные вариации эти способов синтезировали основные варианты решения острой экономико-политической задачи современного мира – осуществления безопасного импортозамещения ПО ТКУ. Произведенная оценка результатов позволяет утверждать, что часть вариантов не может быть осуществлено или же не применима. Другая часть теоретически возможна, но требует отдельного исследования. Третья же часть должна работать с применением авторского метода алгоритмизации машинного кода ТКУ [9, 10, 11, 12, 13].

Авторский метод позволяет восстанавливать архитектуру и логику работы ПО с дальнейшим поиском уязвимостей в нем. При этом он не старается решить невыполнимую задачу – получение компилируемого исходного кода, отличаясь этим от большинства альтернативных. Благодаря разработанным моделям [14, 15, 16, 17, 18, 19], восстанавливаемое представление подходит для анализа экспертом – как с целью понимания архитектуры и алгоритмов работы кода, так и поиска в нем уязвимостей. Базовое тестирование программной реализации прототипа утилиты [20, 21, 22, 23, 24, 25], используемой методом, доказало работоспособность последнего в интересах решаемой задачи. Специально разработанные авторские методики [26, 27] позволяют оценить эффективность работы метода, как достаточно высокую.

Литература

1. Израилов К. Е. Анализ состояния в области безопасности программного обеспечения // Актуальные проблемы инфотелекоммуникаций в науке и образовании. Сборник научных статей II Международной научно-технической и научно-методической конференции. 2013. С. 874–877.

2. Израилов К. Е. Функциональный модуль автоматизированной методики оценки выполнения требований обеспечения безопасности сети связи // Актуальные проблемы информационной безопасности: сборник научных трудов / Отв. редактор Е. В. Стельмашонок. СПб.: СПбГИЭУ, 2012. С. 219–225.

3. Буйневич М. В., Владыко А. Г., Доценко С. М., Симонина О. А. Организационно-техническое обеспечение устойчивости функционирования и безопасности сети связи общего пользования. СПб.: СПбГУТ, 2013. 144 с.

4. Израилов К. Е. Архитектурные уязвимости программного обеспечения // Шестой научный конгресс студентов и аспирантов ИНЖЭКОН-2013: сборник тезисов докладов научно-практической конференции факультета информационных систем и экономике и управлении СПбГИЭУ. 2013. С. 13.

5. Буйневич М. В., Щербаков О. В., Владыко А. Г., Израилов К. Е. Архитектурные уязвимости моделей телекоммуникационных сетей // Научно-аналитический журнал «Вестник Санкт-Петербургского университета Государственной противопожарной службы МЧС России». 2015. № 4. С. 86–93.

6. Буйневич М. В., Израилов К. Е., Мостович Д. И. Ярошенко А. Ю. Проблемные вопросы нейтрализации уязвимостей программного кода телекоммуникационных устройств // Проблемы управления рисками в техносфере. 2016. № 3(39). С. 81–89.

7. Buinevich M., Izrailov K., Vladyko A. The life cycle of vulnerabilities in the representations of software for telecommunication devices **//** 18th International Conference on Advanced Communications Technology (ICACT). 2016. pp. 430–435.

8. Буйневич М. В., Израилов К. Е., Мостович Д. И. Сравнительный анализ подходов к поиску уязвимостей в программном коде // Актуальные проблемы инфотелекоммуникаций в науке и образовании: сборник научных статей V Международной научно-технической и научно-методической конференции. 2016. С. 256–260.

9. Израилов К. Е. Алгоритмизация машинного кода телекоммуникационных устройств как стратегическое средство обеспечения информационной безопасности // Национальная безопасность и стратегическое планирование. 2013. № 2 (2). С. 28–36.

10. Буйневич М. В., Израилов К. Е. Метод алгоритмизации машинного кода телекоммуникационных устройств // Телекоммуникации. 2012. № 12. С. 2–6.

11. Buinevich M., Izrailov K. Method and utility for recovering code algorithms of telecommunication devices for vulnerability search // 16th International Conference on Advanced Communication Technology (ICACT). 2014. pp. 172–176.

12. Buinevich M., Izrailov K., Vladyko A. Method for partial recovering source code of telecommunication devices for vulnerability search // 17th International Conference On Advanced Communications Technology (ICACT). 2015. pp. 76–80.

13. Buinevich M., Izrailov K., Vladyko A. Method and prototype of utility for partial recovering source code for low-level and medium-level vulnerability search // 18th International Conference on Advanced Communication Technology (ICACT). 2016. pp. 700–707.

14. Израилов К. Е. Применение диаграммы Насси-Шнейдермана для анализа структурированных алгоритмов // Пятый научный конгресс студентов и аспирантов ИНЖЭКОН-2012: сборник тезисов докладов научно-практической конференции факультета информационных систем и экономике и управлении СПбГИЭУ. 2012. С. 49–50.

15. Израилов К. Е. Модель прогнозирования угроз телекоммуникационной системы на базе искусственной нейронной сети // Вестник ИНЖЭКОНа. Серия: Технические науки. 2012. № 8 (59). С. 150–153.

16. Израилов К. Е., Васильева А. Ю. Язык описания модели безопасности телекоммуникационной сети // Новые информационные технологии и системы (НИТИС-2012): сборник научных статей X Международной научно-технической конференции. 2012. С. 272–275.

17. Израилов К .Е. Расширение языка «С» для описания алгоритмов кода телекоммуникационных устройств // Информационные технологии и телекоммуникации. 2013. № 2 (2). С. 21–31.

18. Буйневич М. В., Щербаков О. В., Израилов К. Е. Модель машинного кода, специализированная для поиска уязвимостей // Вестник Воронежского института ГПС МЧС России. 2014. № 2 (11). С. 46–51.

19. Буйневич М. В., Щербаков О. В., Израилов К. Е. Структурная модель машинного кода, специализированная для поиска уязвимостей в программном обеспечении автоматизированных систем управления // Проблемы управления рисками в техносфере. 2014. № 3 (31). С. 68–74.

20. Буйневич М. В., Израилов К. Е. Автоматизированное средство алгоритмизации машинного кода телекоммуникационных устройств // Телекоммуникации. 2013. № 6. С. 2–9.

21. Израилов К. Е. Внутреннее представление прототипа утилиты для восстановления кода // Фундаментальные и прикладные исследования в современном мире. 2013. № 2. С. 79–90.

22. Израилов К. Е. Утилита восстановления алгоритмов работы машинного кода // Свидетельство о государственной регистрации программы для ЭВМ № 2013618433 от 09.09.2013.

23. Буйневич М. В., Израилов К. Е. Утилита для поиска уязвимостей в программном обеспечении телекоммуникационных устройств методом алгоритмизации машинного кода. Часть 1. Функциональная архитектура // Информационные технологии и телекоммуникации. 2016. Т. 4. № 1. С. 115–130.

24. Израилов К. Е. Утилита для поиска уязвимостей в программном обеспечении телекоммуникационных устройств методом алгоритмизации машинного кода. Часть 2. Информационная архитектура // Информационные технологии и телекоммуникации. 2016. Т. 4. № 2. С. 86–104.

25. Buinevich M., Izrailov K., Vladyko A. Testing of Utilities for Finding Vulnerabilities in the Machine Code of Telecommunication Devices // 19th International Conference on Advanced Communication Technology (ICACT), 2017, pp. 408–414.

26. Израилов К. Е., Васильева А. Ю., Рамазанов А. И. Укрупненная методика оценки эффективности автоматизированных средств, восстанавливающих исходный код в целях поиска уязвимостей // Вестник ИНЖЭКОНа. Серия: Технические науки. 2013. № 8 (67). С. 107–109.

27. Израилов К. Е. Методика оценки эффективности средств алгоритмизации, используемых для поиска уязвимостей // Информатизация и связь. 2014. № 3. С. 44–47.

References

1. Izrailov, K. Analysis of the Security State of Software // II International Scientific-Technical and Scientific-Methodical Conference «Actual Problems of Education in Science and Education». SPb: SPbSUT. 2013. pp. 874–877.

2. Izrailov, K. E. Function Module of Automated Methodology for Assessing the Implementation of Secure Communications Network Requirements. SPb: SPbGIEU. 2012. pp. 219–225.

3. Buinevich, M. V., Vladyko A. G., Dotsenko S. M., Simonina O. A. Organizational and Technical Provision of Functioning and Security of Public Communications Network. SPb: SPbGUT. 2013. 144 p.

4. Izrailov, K. E. Architectural Software Vulnerabilities // The Sixth Scientific Congress of Students and Graduate Students of ENGECON-2013. SPb: SPbGIEU. 2013. p. 13.

5. Buinevich, M. V., Shcherbakov O. V., Vladyko A. G., Izrailov K. E. Architectural Vulnerability of Telecommunications Networks Models // Vestnik Sankt-Peterburgskogo universiteta Gosudarstvennoy protivopozharnoy sluzhby EMERCOM of Russia. 2015. No. 4. pp. 86–93.

6. Buinevich, M. V., Izrailov K. E., Mostovich D. I. Yaroshenko A. Yu. A Comparative Analysis of Approaches to Finding Vulnerabilities in Software Code // Problemy upravleniya riskami v tekhnosfere. 2016. No. 3 (39). pp. 81–89.

7. Buinevich, M., Izrailov, K., Vladyko, A. The life cycle of vulnerabilities in the representations of software for telecommunication devices // 18th International Conference on Advanced Communications Technology (ICACT). 2016. pp. 430–435.

8. Buinevich, M. V., Izrailov, K. E., Mostovich, D. I. A Comparative Analysis of Approaches to Finding Vulnerabilities in Software Code // V International Scientific-Technical and Scientific-Methodical Conference «Actual Problems of Education in Science and Education». SPb: SPbSUT. 2016. pp. 256–260.

9. Izrailov, K. E. Algorithmization Machine Code of Telecommunication Devices as a Strategic Means of Ensuring the Information Security // Natsional'naya bezopasnost' i strategicheskoe planirovanie. 2013. No. 2 (2). pp. 28–36.

10. Buinevich, M. V., Izrailov, K. E. Method of Algorithmization Machine Code of Telecommunication Devices // Telecommunications. 2012. No. 12. pp. 2–6.

11. Buinevich, M., Izrailov, K. Method and utility for recovering code algorithms of telecommunication devices for vulnerability search // 16th International Conference on Advanced Communication Technology (ICACT). 2014. pp. 172–176.

12. Buinevich, M., Izrailov, K., Vladyko, A. Method for partial recovering source code of telecommunication devices for vulnerability search // 17th International Conference on Advanced Communications Technology (ICACT). 2015. pp. 76–80.

13. Buinevich, M., Izrailov K., Vladyko, A. Method and prototype of utility for partial recovering source code for low-level and medium-level vulnerability search // 18th International Conference on Advanced Communication Technology (ICACT). 2016. pp. 700–707.

14. Izrailov, K. E. The Use of Nassi-Shneiderman Diagrams for Structured Analysis Algorithms // The Fifth Scientific Congress of Students and Graduate Students of ENGECON-2012. SPb: SPbGIEU. 2012. pp. 49–50.

15. Izrailov, K. E. Model of Forecasting the Telecommunication System Threats based on the Artificial Neural Network // Vestnik INGECONa. Seriya: Tekhnicheskie nauki. 2012. № 8 (59). pp. 150–153.

16. Izrailov K. E., Vasilieva A. Yu. Description Language of Telecommunications Network Security Model // Proceedings of the X International Scientific and Technical Conference "New Information Technologies and Systems (NITIS-2012)". 2012. pp. 272–275.

17. Izrailov K. E. C-Language Extension for Algorithm Description of Telecommunication Devices Code // Telecom IT. 2013. No. 2 (2). pp. 21–31. URL: http://www.sut.ru/doci/nauka/review/2-13.pdf

18. Buinevich M. V., Izrailov K. E., Shcherbakov O. V. Model of Machine Code Specialized for Vulnerabilities Search // Vestnik Voronezhskogo instituta GPS of EMERCOM of Russia. 2014. No. 2 (11). pp. 46–51.

19. Buinevich M. V., Shcherbakov O. V., Izrailov K. E. Structural Model of Machine Code Specialized for Search for Software Vulnerabilities of the Automated Control Systems // Problemy upravleniya riskami v tekhnosfere. 2014. No. 3 (31). pp. 68–74.

20. Buinevich M. V., Izrailov K. E. Automated Utility of Algorithmization Machine Code of Telecommunication Devices // Telecommunikacii. 2013. No. 6. pp. 2–9.

21. Izrailov K. E. The Internal Representation of a Prototype Utility for the Recovery Code // Fundamental'nyye i prikladnyye issledovaniya v sovremennom mire. 2013. No. 2. pp. 79–90.

22. Izrailov K. E. Utility for Recovering Machine Code Algorithms // Svidetel'stvo o gosudarstvennoy registratsii programmy dlya EVM № 2013618433. 09.09.2013.

23. Buinevich M. V., Izrailov K. E. Utility for Vulnerability Search in Software of Telecommunication Devices by Method Algorithmization of Machine Code. Part 1. Functional Architecture // Telecom IT. 2016. Vol. 4. No. 1. pp. 115–130. URL: http://sut.ru/doci/nauka/review/20161/115-130.pdf

24. Izrailov K. E. Utility for Vulnerability Search in Software of Telecommunication Devices by Method Algorithmization of Machine Code. Part 2. Information Architecture // Telecom IT. 2016. Vol. 4. No. 2. pp. 86–104. URL: http://sut.ru/doci/nauka/review/20162/86-104.pdf

25. Buinevich M., Izrailov K., Vladyko A. Testing of Utilities for Finding Vulnerabilities in the Machine Code of Telecommunication Devices // 19th International Conference on Advanced Communication Technology (ICACT), 2017, pp. 408–414.

26. Izrailov K. E., Vasilieva A. Yu., Ramazanov A. I. Enlarged Assessment Method of Effectiveness of Automated Tools, Recovering Source Code in Search for Vulnerability // Vestnik INGECONa. Seriya: Tekhnicheskie nauki. 2013. No. 8 (67). pp. 107–109.

27. Izrailov K. E. Methods of Assessing the Effectiveness of Medium of Algorithmization Used to Find Vulnerabilities // Informatsiya I svyaz'. 2014. No. 3. pp. 44–47.

Буйневич Михаил Викторович – доктор технических наук, профессор, ведущий научный сотрудник, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, bmv1958@yandex.ru

Израилов Константин Евгеньевич – аспирант, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, konstantin.izrailov@mail.ru

Buinevich Michael – D.Sc., professor, leading scientific researcher, SPbSUT, St. Petersburg, 193232, Russian Federation, bmv1958@yandex.ru

Izrailov Konstantin – postgraduate, SPbSUT, St. Petersburg, 193232, Russian Federation, konstantin.izrailov@mail.ru