ПРОТИВОДЕЙСТВИЕ УЯЗВИМОСТЯМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. Часть 1. ОНТОЛОГИЧЕСКАЯ МОДЕЛЬ

Леонов Н. В.[1]

DOI: 10.21681/2311-3456-2024-2-87-92

Цель исследования: *концептуальное противодействие уязвимостям в программном обеспечении.*

Методы исследования: *системный анализ, моделирование, разработка тематической онтологии.*

Полученные результаты: *в первой части статьи введена онтологическая модель области программного обеспечения с уязвимостями, взаимоувязывающая ее основные сущности: субъектов – разработчика программного обеспечения, внедряющего уязвимости нарушителя, эксперта по их обнаружению и нейтрализации; и используемых ими объекты – задачу, программу, исходный и исполняемый код, вектор атаки, уязвимость, отчет безопасности, патч, мета-информацию, а также средства сборки, сканирования, внедрения, обфускации и реверс-инжиниринг.*

Научная новизна *работы определяется охватом сразу трех направлений предметной области (программный инжиниринг, внедрение уязвимостей и их нейтрализация), а также использовании строгих правил (или шаблонов) связей между ее сущностями.*

Ключевые слова: *информационная безопасность, уязвимость, противодействие, онтологическая модель.*

COUNTERING SOFTWARE VULNERABILITIES. Part 1. ONTOLOGICAL MODEL

Leonov N. V.[2]

The goal of the investigation: *conceptual counteraction to software vulnerabilities.*

Research methods: *system analysis, modeling, development of thematic ontology.*

Results: *in the first part of the article, an ontological model of the software domain with vulnerabilities is introduced, interrelating its main entities: subjects - a software developer who introduces an attacker's vulnerabilities, an expert in their detection and neutralization; and the objects they use - task, program, source and executable code, attack vector, vulnerability, security report, patch, meta-information, as well as scanning, injection, obfuscation and reverse engineering tools.*

The scientific novelty *of the work is determined by the coverage of three areas of the subject area at once (software engineering, the introduction of vulnerabilities and their neutralization), as well as the use of strict rules (or templates) for connections between its entities.*

Keywords: *information security, vulnerability, counteraction, ontological model*

Введение

Использование небезопасного программного обеспечения (далее – ПО) является актуальнейшей проблемой сферы информационных технологий. Одна из причин угрозы со стороны ПО заключается в наличии уязвимостей в конечном продукте (т.е. программе или их группе), внесенных туда на одном из этапов программного инжиниринга – при создании его архитектуры, алгоритмов, исходного кода или даже в процессе его сборки в исполняемый. Наиболее трудной с точки зрения противодействия считается ситуация, когда уязвимость вносилась намеренно нарушителем (далее – Нарушитель), поскольку в этом случае она может сознательно быть скрыта от обнаружения (например, путем реализации через

1 Леонов Николай Викторович, кандидат технических наук, доцент, начальник лаборатории Государственного научно-исследовательского института прикладных проблем. ORCHID: http://orcid.org/0009-0005-1295-5343. E-mail: leonov-nv@yandex.ru

2 Nikolay V. Leonov, Ph.D., assistant Professor, Head of the State Research Institute of Applied Problems Laboratory. ORCHID: http://orcid.org/0009-0005-1295-5343. E-mail: leonov-nv@yandex.ru

код, сигнатура которого не совпадает с имеющимися в базах данных антивирусов) [1]. При этом сам код уязвимости не всегда может быть отделен от легального, поскольку для его разработки используются стандартный инструментарий, парадигмы программирования, средства сборки, инструкции процессора и т.п. Таким образом, процесс противодействия уязвимостям оказывается крайне сложным из-за нетривиальной зависимости от множества внешних факторов, поскольку различные его действия могут как позитивно, так и негативно влиять на итоговый результат [2]; например, усложнение анализа кода, с одной стороны, затруднит внедрение туда уязвимостей, а с другой – их обнаружение. Такое противодействие между экспертом-«безопасником» и Нарушителем в рамках одного программного продукта можно представить в виде шахматной партии, где доской является код программы, фигурами – конструкции программы, а целью игры (т.е. матом в классическом понимании) – обнаружение или сокрытие уязвимости. Данное противостояние может быть распространено и на всю предметную область программ с уязвимостями с позиции процессов их создания, внедрения, развития, обнаружения и нейтрализации. Оценка и повышение эффективности последних двух процессов может быть достигнута только глубоким пониманием всех сущностей предметной области и их взаимосвязей, а также действующих в предметной области законов [3]. В интересах этого далее будет предложена ее онтологическая модель, а также дан ряд концептуальных выводов и прогнозов с целью повышения информационной безопасности ПО.

Онтологическая модель

Предложим онтологическую модель предметной области (далее – Модель) в форме графической схемы, как совокупности ее элементов – т.е. сущностей и их взаимосвязей, отражающих любую программу в аспекте ее создания, внедрения уязвимостей и их нейтрализации. Опишем виды элементов Модели с указанием используемой формы, цветовой раскраски и примеров:

1) пассивный объект (далее – П-объект), являющийся некоторой совокупностью информации, которая может быть преобразована в другую или подвержена изменению внешним алгоритмом (прямоугольник), и который подразделяется на исходный (желтый), производный промежуточный (зеленый) и производный конечный (оранжевый) П-объект – например, Программа;
2) активный объект (далее – А-объект), обладающий заданным алгоритмом преобразования П-объектов (прямоугольник синего фона) – например, Средство обфускации;
3) субъект, обладающий разумом и способный продуцировать новые (т.е. заранее не заданные) алгоритмы преобразования П-объектов или управления другими преобразованиями (фигура человека) – например, Эксперт;
4) взаимосвязь, показывающая отношения между сущностями (стрелка):
   - преобразование объекта в другой или его изменение согласно внешнему алгоритму (объемная); например, получение Отчета безопасности из Программы, используя алгоритм Средства сканирования;
   - применение алгоритма, заложенного в А-объект, для управления преобразованием П-объектов (сплошная, к объемной); например, Средство сканирования применяет свой алгоритм по поиску Уязвимостей, преобразовывая Программу в Отчет безопасности;
   - применение алгоритма, продуцированного субъектом, для преобразования П-объектов (пунктирная, к объемной); например, Нарушитель продуцирует собственный алгоритм для создания Уязвимости (в виде концептуальной идеи, архитектуры или ее алгоритмов), исходя из будущего Вектора атак (т.е. по заданному Вектору атак создаются уязвимости, которые необходимо внедрить для его успешного осуществления);
   - применение алгоритма, продуцированного субъектом, для управления другими преобразованиями (пунктирная, к сплошной); например, Эксперт продуцирует собственный алгоритм для управления применением Средства сканирования, настраивая в том числе параметры и точки приложения последнего;
   - использование субъектом информации из объекта для продуцирования в будущем алгоритмов (пунктирная, к фигуре человека); например, Эксперт использует Мета-информацию для понимания работы Программы и поиска в ней Уязвимостей.

Также в Модели присутствует 3 прямоугольника с закругленными краями и красной штриховкой, соответствующие направлениям предметной области, о которых будет сказано далее.

Используя указанные формы Модели, удалось значительно упростить представление Модели, поскольку все ее взаимосвязи не имеют субъективных свойств (т.е. «классических» надписей на стрелках), а строго формализованы.

Также в названиях использована метка-обозначение в постфиксе названий «(*)» – тождественность двух фигур, соответствующих одной сущности

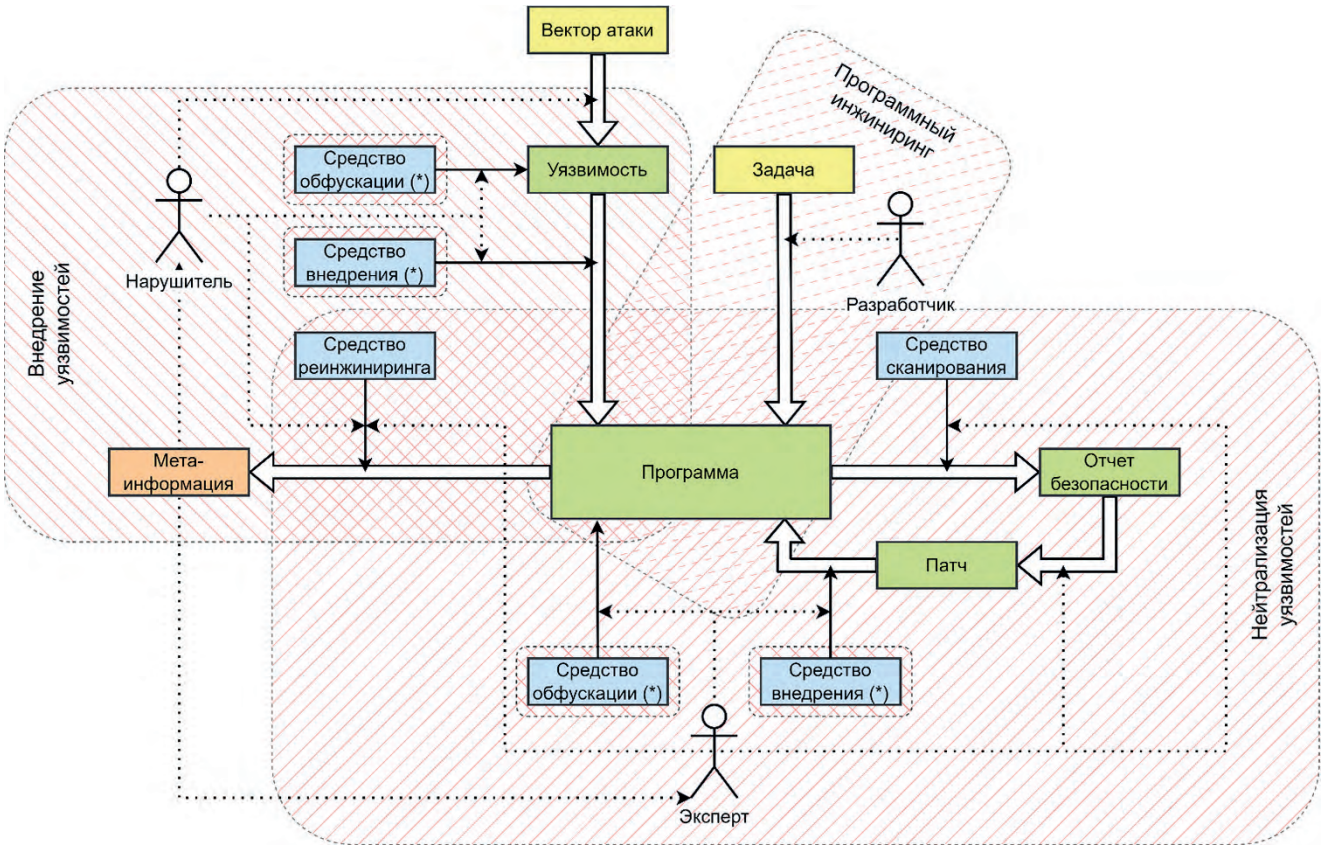

Рис. 1. Онтологическая модель предметной области

(используется для разнесения фигур с целью незагромождения рисунка; суть метки будет разъяснена далее.

Модель в схематичном виде, согласно введенным обозначениям, представлена на рис. 1.

Модель (см. рис. 1) состоит из следующих сущностей (которые здесь и далее пишутся с прописной буквы) и взаимосвязей:

а) субъекты в количестве 3 штук:
   1. Разработчик – как правило, сотрудник IT-компании, который проектирует (в широком смысле: замысел, алгоритмизация, программирование, отладка) Программу для решения определенной Задачи;
   2. Нарушитель – злоумышленник, который создает Уязвимость для внедрения в Программу с целью реализации угроз информационной безопасности (согласно Вектору атак) [4].
   3. Эксперт – специалист в области информационной безопасности, который стремится обеспечить безопасность использования Программы за счет нейтрализации Уязвимостей или препятствования их внедрению (т.е. занимается противодействием Нарушителю) [5];

б) П-объекты в количестве 7 штук:
   1. Задача – цель-ориентированная проблемная ситуация, решение которой должно быть осуществлено Разработчиком с применением информационной системы; данный объект является одним из инициаторов всех преобразований;
   2. Программа – совокупность программного кода, данных и служебной информации (как в предварительном, так и готовом виде) для управления информационной системой, обладающая функционалом по решению поставленной Задачи (при необходимости, декомпозируемой на несколько подзадач); под данной сущностью будем понимать легальную Программу, т.е. изначально без Уязвимостей; данный объект является центральным в Модели;
   3. Вектор атаки – направление действий Нарушителя для реализации информационных угроз в информационной системе; данный объект является одним из инициаторов всех преобразований;
   4. Уязвимость – программный дефект в виде Исходного кода, созданного Нарушителем, эксплуатация которого может привести к реализации угрозы (атаке); необходимо уточнить, что случайные дефекты не рассматриваются;
   5. Отчет безопасности – информация об обнаруженных в Программе Уязвимостях (в том числе, содержащая их метрики, области кода и т.п.) полученная Экспертом (в том числе,

из внешних специализированных организаций) при помощи Средства сканирования [6];
6. Патч (другое название – «заплатка») – некоторый код, который внедряется в Программу с целью устранения Уязвимости (путем замены ее кода или нейтрализации ее эффекта) [7]; очевидно, создается и применяется Экспертами;
7. Мета-информация – информация о Программе, отражающая принципы и детали ее работы (например, компоненты, модули, подпрограммы, логику и т.п.) [8];

в) А-объекты в количестве 4 штук, из которых 2 дублируются:
1. Средство сканирования – программное средство, используемое для обнаружения Уязвимостей в Программе (путем ее анализа, например, с применением сравнения сигнатур кода) и генерации по результатам соответствующего Отчета безопасности [9]; под данным средством, в основном используемым Экспертом, понимаются как сложные и имеющие высокую степень автоматизации (например, антивирусы), так и тривиальные, требующие существенного участия человека (например, парсеры машинного или байт-кода, результаты которого необходимо обрабатывать вручную);
2. Средство внедрения – программное средство для помещения специализированного кода в Программу (в любое из ее представлений); при этом средство может использоваться Нарушителем – для внедрения Уязвимостей [10,11], и Экспертом – для применения Патча; метка «(*)» означает, что Средства внедрения могут состоять из одного набора утилит;
3. Средство обфускации – программное средство для запутывания кода Программы (или ее любого представления) с целью затруднения последующего анализа и восстановления из нее Мета-информации [12]; средство может использоваться как Нарушителем – для затруднения обнаружения «своих» Уязвимостей (например, сигнатурным поиском [13]), так и Экспертом – для затруднения внедрения Уязвимостей, поскольку последние должны быть корректно встроены в готовое решение; метка «(*)» означает, что Средства обфускации могут состоять из одного набора;
4. Средство реинжиниринга (сокр. от реверс-инжиниринг) – программное средство, позволяющее анализировать Программу для восстановления из нее Мета-информации [8]; средство необходимо как Нарушителю – для определения мест и механизмов внедрения Уязвимостей, так и Эксперту – для их обнаружения и написания Патчей; классическими примерами средств являются дизассемблеры, декомпиляторы [14], построители блок-схем, анализаторы программных взаимодействий [15] и т.п.;

г) взаимосвязи с участием трех сущностей в количестве 7 штук (в форме «А-объект/Субъект ⟶ (Исходный П-объект ⇒ Конечный П-объект)»):
1. «Разработчик ⟶ (Задача ⇒ Программа)» – ручной анализ Разработчиком поставленной Задачи с целью продуцирования алгоритма по созданию Программы ее решения;
2. «Эксперт ⟶ (Отчет безопасности ⇒ Патч)» – ручной анализ Экспертом сформированного Отчета безопасности с целью продуцирования алгоритма по созданию Патча к Программе для устранения указанных в отчете уязвимостей;
3. «Нарушитель ⟶ (Вектор атаки ⇒ Уязвимость)» – ручной анализ Нарушителем Вектора атаки с целью продуцирования алгоритма по созданию Уязвимости, внедрение которой в Программу позволить реализовывать информационные угрозы;
4. «Средство сканирования ⟶ (Программа ⇒ Отчет безопасности)» – автоматизированное формирование Отчета безопасности по результатам анализа Программы с помощью Средства сканирования;
5. «Средство внедрения (*) ⟶ (Патч ⇒ Программа)» – автоматизированное изменение Программы путем применения к ней Патча (для устранения Уязвимости) с помощью Средства внедрения (производится Экспертом для повышения безопасности Программы);
6. «Средство внедрения (*) ⟶ (Уязвимость ⇒ Программа)» – автоматизированное изменение Программы путем «вложения» в нее Уязвимости с помощью Средства внедрения (производится Нарушителем для снижения безопасности Программы);
7. «Средство реинжиниринга ⟶ (Программа ⇒ Мета-информация)» – автоматизированный анализ Программы для восстановления из нее Мета-информации с помощью Средства реинжиниринга, производимый следующими субъектами для повышения эффективности их алгоритмов: Экспертом – при поиске Уязвимостей, создании и внедрении Патча, обфускации Программы и ее вторичном реинжиниринге; Нарушителем – при создании Уязвимости, ее обфускации и внедрении, а также вторичном реинжиниринге Программы;

д) взаимосвязи с участием двух сущностей в количестве 2 штук (в форме «Источник ⟶ Получатель»):
1. «Мета-информация ⟶ Эксперт» – получение Экспертом восстановленной Мета-информации для ее анализа в интересах повышения безопасности Программы;

2. «Мета-информация → Нарушитель» – получение Нарушителем восстановленной Мета-информации для ее анализа в интересах снижения безопасности Программы;

е) взаимосвязи с участием двух сущностей и другой взаимосвязи в количестве 7 штук (в форме «Субъект → (А-объект → |)»):

1. «Эксперт → (Средство сканирования → |)») – продуцирование Экспертом алгоритма управления действиями Средства сканирования (например, его применение для особо критичных областей Программы);
2. «Эксперт → (Средство внедрения (*) → |)») – продуцирование Экспертом алгоритма управления действиями Средства внедрения (например, указание параметров применения Патча);
3. «Эксперт → (Средство обфускации (*) → |)») – продуцирование Экспертом алгоритма управления действиями Средства обфускации (например, указание количества проходов обфускации);
4. «Эксперт → (Средство реинжиниринга → |)») – продуцирование Экспертом алгоритма управления действиями Средства реинжиниринга (например, корректировка его работы по восстановлению потока управления);
5. «Нарушитель → (Средство внедрения (*) → |)») – продуцирование Нарушителем алгоритма управления действиями Средства внедрения (например, указание позиции для помещения Уязвимости в Программу);
6. «Нарушитель → (Средство обфускации (*) → |)») – продуцирование Нарушителем алгоритма управления действиями Средства обфускации (например, выбор метода обфускации);
7. «Нарушитель → (Средство реинжиниринга → |)») – продуцирование Нарушителем алгоритма управления действиями Средства реинжиниринга (например, корректировка его работы по восстановлению потока данных).

В Модели присутствуют следующие 3 вышеупомянутых направления в предметной области (помечены областью с красной штриховкой).

Первое направление – «Программный инжиниринг», группирующее сущности по разработке задача-ориентированной Программы. Данный процесс, являющийся линейным, выполняется Разработчиком и состоит из одного этапа – Разработчик анализирует поставленную Задачу и создает код (Исходный, преобразуемый в Исполняемый) Программы для ее решения.

Второе направление – «Нейтрализация уязвимостей», группирующее сущности для обеспечения безопасности Программы путем нейтрализации ее Уязвимостей. Данный процесс, являющийся циклическим, выполняется Экспертом и состоит из следующих этапов. Во-первых, Эксперт, используя Средство сканирования, анализирует Программу и составляет Отчет безопасности о найденных Уязвимостях (в ручном или автоматическом режиме). Во-вторых, используя данные из отчета, Эксперт создает Патч для повышения безопасности исходной Программы путем его внедрения с помощью соответствующего Средства. В-третьих, Экспертом дополнительно может применяться Средство обфускации для усложнения анализа Программы Нарушителем (при внедрении им Уязвимостей). Затем цикл повторяется, поскольку может происходить как эволюционирование текущей Уязвимости [16], так и появляться новые. Также на всех этапах процесса Эксперту может требоваться понимание принципов и деталей работы Программы, для чего требуется восстановление Мета-информации с помощью Средства реинжиниринга.

Третье направление – «Внедрение уязвимостей», группирующее сущности для снижения безопасности Программы путем внедрения в нее Уязвимостей [17]. Данный процесс, также являющийся циклическим, выполняется Нарушителем и состоит из следующих этапов. Во-первых, согласно изначально имеющемуся Вектору атаки, Нарушитель создает Уязвимость. Во-вторых, Нарушитель может применять Средство обфускации для затруднения обнаружения Уязвимости соответствующим Средством сканирования. В-третьих, Уязвимость помещается в Программу с помощью специализированного Средства внедрения. И, в-четвертых, после выпуска Патчей по нейтрализации Уязвимости, она модифицируется, что приводит к повторению цикла. Как и Эксперту, на всех этапах процесса Нарушителю также может требоваться восстановление Мета-информации с помощью Средства реинжиниринга.

Программа, являясь центральной и комплексной сущностью Модели, сама может быть представлена в виде отдельной вложенной онтологической подмоделью, показанной на рис. 2, которая состоит из трех

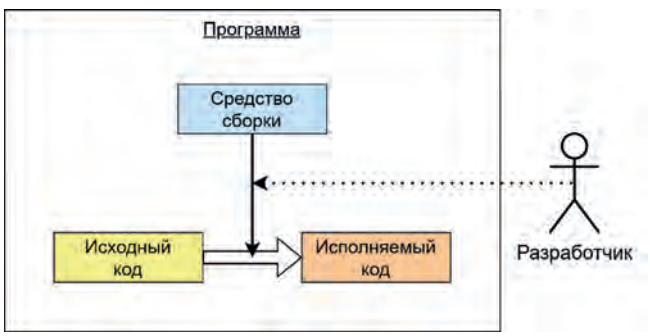

Рис. 2. Онтологическая модель программы для информационной системы

следующих объектов (помимо введённого ранее субъекта):

1. Исходный код – формализованное описание Программы в текстовой или графической форме (как на языке программирования, так и с помощью блок-схем алгоритмов или архитектуры [18]), и подходящее субъектам для работы с ним (исходным кодом);
2. Исполняемый код – формализованное описание Программы в бинарном виде, подходящее для выполнения в информационной системе (т.е. имеющее вид машинных инструкций или байт-кода) [19];
3. Средство сборки – программное средство для преобразования человеко-ориентированного описания программы – Исходного кода, в машинно-ориентированное – Исполняемый код (т.е. с применением последовательности утилит компиляции, ассемблирования, линковки и др.) [20].

Также в данной модели присутствуют следующие две взаимосвязи:

1) с участием трех сущностей (в форме «А-объект ⟶ (Исходный П-объект ⇒ Конечный П-объект)»): «Средство сборки ⟶ (Исходный код ⇒ Исполняемый код)» – сборка Исполняемого кода Программы из Исходного;
2) с участием двух сущностей и другой взаимосвязи (в форме «Субъект ⟶ (А-объект ⟶ |)»): «Разработчик ⟶ (Средство сборки ⟶ |)») – продуцирование Разработчиком алгоритма управления действиями Средства сборки (например, выбор уровня оптимизации Программы).

Необходимо отметить, что формально Средства сборки используются помимо Разработчика также Нарушителем и Экспертом, однако в случае двух последних данные средства носят необязательный характер (т.к. зачастую данными субъектами может разрабатываться ассемблерный код) или же являются лишь отдельными утилитами полноценных средств (например, свободно распространяемые или собственного производства компиляторы).

Продолжение следует ...

Литература

1. *Леонов Н. В., Буйневич М. В. Проблемные вопросы поиска уязвимостей в программном обеспечении промышленных ИТ-устройств // Автоматизация в промышленности. 2023. № 12. С. 59–63.*
2. *Леонов Н. В., Буйневич М. В. Машинное обучение VS поиск уязвимостей в программном обеспечении: анализ применимости и синтез концептуальной системы // Труды учебных заведений связи. 2023. Т. 9. № 6. С. 83–94. DOI: 10.31854/1813-324X-2023-9-6-83-94.*
3. *Васильев В. И., Вульфин А. М., Кучкарова Н. В. Автоматизация анализа уязвимостей программного обеспечения на основе технологии Text Mining // Вопросы кибербезопасности. 2020. № 4 (38). С. 22–31. DOI: 10.21681/2311-3456-2020-04-22-31.*
4. *Лукацкий А. В. Обзор мировых трендов по промышленной кибербезопасности // Релейщик. 2020. № 1 (36). С. 60–62.*
5. *Вареница В. В., Марков А. С., Савченко В. В., Цирлов В. Л. Практические аспекты выявления уязвимостей при проведении сертификационных испытаний программных средств защиты информации // Вопросы кибербезопасности. 2021. № 5 (45). С. 36–44. DOI: 10.21681/2311-3456-2021-5-36-44.*
6. *Якимук А. Ю., Устинов С. А., Лазарев Т. П., Коваленко А. С. Методы формализации описания сценариев кибератак // Электронные средства и системы управления. Материалы докладов Международной научно-практической конференции. 2022. № 1–2. С. 73–76.*
7. *Коржев А. А. Обеспечение безопасности программного обеспечения // Стратегическое развитие инновационного потенциала отраслей, комплексов и организаций: сборник статей XI Международной научно-практической конференции (Пенза, 10–11 октября 2023 года). 2023. – С. 237–241.*
8. *Израилов К. Е. Методология реверс-инжиниринга машинного кода. Часть 2. Статическое исследование. Труды учебных заведений связи // 2023. Т. 9. № 6. С. 68–82. DOI: 10.31854/1813-324X-2023-9-6-68-82.*
9. *Суздалов Д. В., Некрасов А. Н. Разработка сканера уязвимостей // Наука молодых: сборник материалов Межрегиональной молодежной научной конференции, посвященной памяти Ф. А. Бабушкина, (Сыктывкар, 25–26 мая 2023 года). 2023. С. 139–143.*
10. *Руднев Н. О., Герасимова В. Ф., Шагапов И. А. Метод закрепления доступа в системе посредством инъекции кода в операционной системе Windows // Естественные и технические науки. 2022. № 12 (175). С. 398–403.*
11. *Нефедов В. В. Методы внедрения кода в исполняемые файлы PE-формата // Молодежная научная школа кафедры «Защищенные системы связи». 2021. Т. 1. № 2 (4). С. 61–68.*
12. *Градский Д. Ю. Методы обфускации кода // Оригинальные исследования. 2020. Т. 10. № 5. С. 177–180.*
13. *Чукляев И. И., Чепурной Е. А., Шевченко А. Л., Пильненкий В. П. Способы и средства обнаружения и предотвращения информационно-технических воздействий // Системы компьютерной математики и их приложения. 2021. № 22. С. 180–189.*
14. *Маркин Д. О., Макеев С. М. Система защиты терминальных программ от анализа на основе виртуализации исполняемого кода // Вопросы кибербезопасности. 2020. № 1 (35). С. 29–41. DOI: 10.21681/2311-3456-2020-01-29-41.*
15. *Буйневич М. В., Ганов Г. А., Израилов К. Е. Интеллектуальный метод визуализации взаимодействий программ в интересах аудита информационной безопасности операционной системы // Информатизация и связь. 2020. № 4. С. 67–74.*
16. *Израилов К. Е. Моделирование программы с уязвимостями с позиции эволюции ее представлений. Часть 1. Схема жизненного цикла // Труды учебных заведений связи. 2023. Т. 9. № 1. С. 75–93. DOI:10.31854/1813-324X-2023-9-1-75-93.*
17. *Кондаков С. Е., Архипов А. Н. Математическая модель эксплойта, внедренного в файл неисполняемого формата // Известия Института инженерной физики. 2023. № 3 (69). С. 93–95.*
18. *Стецко А. С., Гойник В. А., Набиуллин В. В. Выбор входного языка для графической среды программирования // Электронные средства и системы управления. Материалы докладов Международной научно-практической конференции. 2021. № 1-2. С. 97–99.*
19. *Петухов В. А. Генерация кода для тестирования компиляторов с использованием генеративно-состязательных сетей // Автоматизация в промышленности. 2021. № 6. С. 59–62. DOI: 10.25728/avtprom.2021.06.12.*
20. *Афонин М. В. Компиляция. Сборка и связывание проектов // Инновационный потенциал развития общества: взгляд молодых ученых: сборник научных статей 3-й Всероссийской научной конференции перспективных разработок (Курск, 01 декабря 2022 года). Том 3. 2022. С. 115–118.*