DOI: 10.25728/avtprom.2021.11.09

Способ вариативной классификации уязвимостей в программном коде. Часть 1. Стратификация и категориальное деление

М.В. Буйневич (СПб УГПС МЧС), К.Е. Израилов (СПбГУТ), В.В. Матвеев (СПбГЭУ), В.В. Покусов (КАИБ)

Решается задача вариативной классификации уязвимостей программного кода. В результате обзора релевантных научных работ были установлены границы предметных областей, а также плюсы и минусы известных подходов. Предложен способ классификации, основанный на пяти предпосылках: автоматизация, экспертное мнение, машинное обучение, необходимость и достаточность деления, работа с человеко-ориентированными данными. Указаны шаги по реализации способа, которые позволяют создавать подходящие классы и относить к ним найденные и гипотетические уязвимости.

В первой части статьи на примере программного обеспечения телекоммуникационных устройств описаны первые четыре шага способа, а именно: экспертная стратификация уязвимостей, категориальное деление множества уязвимостей на группы, составление набора классов и предварительное отнесение к ним уязвимостей[1].

Ключевые слова: программное обеспечение телекоммуникационных устройств, уязвимости программного кода, адаптивная классификация, категориальный анализ, машинное обучение.

Введение

Уязвимости в программном обеспечении являются «головной болью» специалистов ИТ-отрасли по всему миру, поскольку их эксплуатация злоумышленниками напрямую влияет на нарушение безопасности информации и может приводить к критическим последствиям как для отдельной личности и бизнес-сообщества, так и для государства в целом.

Для общей оценки безопасности программ (как уже созданных, так и еще только проектируемых) и выбора мер противодействия уязвимостям требуется классификация последних, поскольку с каждым классом могут быть связаны определенные характеристики и метрики безопасности, способы поиска и нейтрализации, возможные источники и уровни угроз, схемы кибератак и т.п.

Уязвимости программного кода являются достаточно разнородными, а кроме того, «контекстными» – зависящими от области появления; например, уязвимости для Web- и desktop-приложений имеют существенные отличия, и использовать для них *единую классификацию* не всегда целесообразно. С другой стороны, по меткому замечанию проф. Розовой С.С. [1], «...качество *конкретных классификаций*, как правило, является неудовлетворительным».

Познание «мира уязвимостей» представляется бесконечно вариативным, поэтому и способ их классификации, как метода научного исследования, призван быть вариативным. Следовательно, первым шагом исследования уязвимостей может стать получение способа их деления на группы. Такой способ позволил бы эксперту, решающему конкретную прикладную задачу в рамках сугубо прагматичной цели, создать собственные классификации, которые учитывали бы особенности его предметной области (как, например это сделано в [2]).

Пошаговой реализации идеи такого способа и посвящена настоящая статья. Авторы считают его прообразом именно того «дефицитного» средства, о котором писала еще свыше 30 лет назад проф. Розова С.С.: «Построение новых классификаций наталкивается на многочисленные и разнообразные трудности, преодоление которых оказывается невозможным из-за отсутствия необходимых теоретических и методических средств» [1].

Аналитический обзор релевантных работ

В доказательство правоты тезиса о многочисленности и разнообразии классификаций уязвимостей в программном коде, а также в интересах формирования требований к реализации способа вариативной классификации уязвимостей в программном коде проведем аналитический обзор научных работ, так или иначе затрагивающих эту область знаний.

Будем оценивать способы на наличие универсальности/специфичности предметной области и человеко/машинности метода решения задачи классификации.

[1] Работа выполнена в рамках реализации проекта AP06851400 «Разработка способа и автоматизация поиска уязвимостей в машинном коде телекоммуникационных устройств» (при финансовой поддержке Министерства образования и науки Республики Казахстан).

Универсальность здесь понимается как некая претензия авторов релевантных работ на абстрактность цели классификации, опирающуюся на псевдоуниверсальные показатели, которые на строгую поверку таковыми не являются. Поэтому в рамках статьи «универсальность» («Ун») считается все же условной.

Согласно [3] классификация является лишь разновидностью традиционного научного исследования в части задачи упорядочения знаний, поэтому обладает всеми его признаками, а именно, наличием объекта, предмета и метода ее решения. Поэтому специфичность (Сп) произведенных классификаций в зависимости от научных предпочтений их авторов и сделанных ими акцентов (то есть определенной ориентации) может выражаться в их объект, предмет и метод ориентированности. Однако в настоящей статье, согласно поставленной научной задаче, интерес представляет специфичность только в ОБЪЕКТЕ, то есть учете специфики программного кода абстрактного объекта или программного обеспечения специфичного объекта.

Человеко/машинность понимается как СУБЪЕКТНАЯ доминанта либо эксперта по безопасности программного кода (Чл – человека), либо программы или алгоритма (Ав – автомата) в реализации метода решения задачи классификации. Могут иметь место и «смеси»: когда часть реализуется экспертом, а часть программно – например, при многоэтапном решении задачи; или когда заявлен аналитический метод, который может исполняться как экспертом, так и программой.

В статье [4] предлагается классификация уязвимостей безопасности информации, основанная на их следующих показателях: фатальность, доступность и число. Показатели оцениваются субъективно с помощью экспертов по балльной системе. В результате вводятся коэффициенты опасности каждой уязвимости, которые представляются в табличном виде.

В работе [5] описаны два альтернативных подхода к классификации уязвимостей программ: при помощи методов дискретной оптимизации и сортировки с применением нечетких чисел. Таким образом, способы классификации по большей части являются аналитическими.

Исследование [6] посвящено автоматическому способу классификации уязвимостей прикладного программного обеспечения. Способ основан на выявлении уязвимостей, обладающих определенным набором признаков. Затем во множестве признаков выделяются отдельные кластеры, которые как раз соответствуют классам и по близости к которым осуществляется разнесение уязвимостей с помощью аппарата нечетной логики.

В работе [7] решает близкую к текущей задача поиска уязвимостей «нулевого дня» путем классификации и разметки текста, содержащего описание таких уязвимостей, в том числе полученных из открытых источников. Предлагаются следующие три принципа, на которых строится решение задачи: наличие словаря слов (например, Зализняка), набор морфологических правил разбора, применение машинного обучения.

Программа для ЭВМ «Компонент классификации уязвимостей на основе их неформальных признаков для выявления слабых мест информационных систем» [свид. о рег. *RU № 2018664027 от 08.11.2018, авт. Дойникова Е.В., Федорченко А.В.*] является частью метасистемы, которая предназначена для реагирования на инциденты информационной безопасности. Функция программы заключается в интеллектуальном анализе открытой информации об уязвимостях (NVD, CWE, CAPEC). Решение основано на применении машинного обучения в части классификации для определения слабых мест систем по их признакам и существующим в базах уязвимостям.

Работа [8] посвящена автоматическому определению критичности уязвимостей, на основе их текстового описания в базе данных NVD. Для этого создается многомерное векторное пространство слов, по которому затем строится модель машинного обучения.

В статье [9] предлагается определять наиболее общие типы уязвимостей сетевых протоколов путем анализа их спецификаций. Показано, что спецификации предусмотрены только для части уязвимостей.

Программа для ЭВМ «Программа классификации уязвимостей автоматизированных систем» [свид. о рег. *RU № 2020617815 от 15.07.2020, авт. Зайченко Я.Б.*] находит соответствие между уязвимостями автоматизированных систем и уязвимостями из существующей базы данных. При этом каждой такой уязвимости присваивается уникальный код.

ПО «Компонент классификации уязвимостей интерфейсов взаимодействия с транспортной инфраструктурой умного города» [свид. о рег. *RU № 2020661231 от 18.09.2020, авт. Израилов К.Е., Виткова Л.А и Чечулин А.А.*], решает задачу автоматической классификации уязвимостей транспортной инфраструктуры с помощью методов машинного обучения в части кластеризации. Описание уязвимостей преобразуется в «мешок слов» [10], в результате чего получается некоторая точка в N-мерном пространстве. Группирование же таких точек может означать их принадлежность к определенному классу.

В [11] применяется аппарат категориального деления, который позволяет выделить заведомо непересекающиеся классы. Суть деления заключается в назначении категориальных пар, элементы которых являются антагонистами по некоторому признаку (например, статический vs динамический, форма vs содержание, алгоритм vs данные). Тем самым, каждая из уязвимостей, во-первых, будет обязательно отнесена к какому-либо классу – условие *достаточности* деления, а, во-вторых, будет отнесена только к одному классу – условие *необходимости* деления. Число классов будет равно 2^P, где P – число категориальных пар.

В статье [12] описывается синтетически полученная обобщенная классификация уязвимостей с использованием дерева рубрикатора. Приводятся некоторые характеристики уязвимостей.

Классификация уязвимостей интегрированных систем управления на ранних стадиях их жизненного цикла рассматривается в [13]. Вводятся четыре критерия: негативное воздействие внешних и внутренних факторов, стадии и процессы жизненного цикла, общность и частность автоматизированных и интегрированных систем, степени

Таблица. Ориентация и реализация методов решения задачи классификации в релевантных работах (в хронологическом порядке)

Релевантные работы		Ориентация		Реализация		Метод решения задачи
Ссылка, свид. о рег.	*Год опубл.*	*«Ун»*	*Сп*	*Чл*	*Ав*	
[12]	2013	+		+		дерево рубрикатора
[15]	2013	+		+		атрибутивное моделирование
[19]	2013	+			+	**машинное обучение**
[20]	2015	+		+		k-ближайших соседей
[4]	2016	+		+		экспертная оценка
[7]	2017	+			+	машинное обучение
[13]	2017		+	+		контекстный анализ
[5]	2017	+		+		дискретная оптимизация и нечеткая логика
[6]	2017	+			+	кластерный анализ и нечеткая логика
RU № 2018664027	2018	+			+	**машинное обучение**
[14]	2018	+		+		количественное и качественное ранжирование
[17]	2019	+			+	**машинное обучение**
[21]	2019	+			+	**машинное обучение**
[8]	2020	+			+	**машинное обучение**
[9]	2020		+	+		контекстный анализ
RU № 2020617815	2020	+			+	поиск соответствия (алгоритм)
RU № 2020661231	2020	+			+	**машинное обучение**
[16]	2020	+		+	+	**машинное обучение**
[18]	2020	+			+	**машинное обучение**
[11]	2021	+		+		категориальный анализ

опасности в случае реализации угрозы (поэтому с каждой уязвимостью в обязательном порядке связывается некоторая угроза). Также уязвимости группируются по способам их нейтрализации или ослабления.

В [14] представлены результаты анализа систем классификации уязвимостей. Выделены три обобщенных способа: количественное и качественное ранжирование, а также применение комплексных показателей. Последний представляется наиболее перспективным, особенно для определения уровня критичности уязвимости.

Различные признаки уязвимостей рассматриваются в [15]. Синтезируется модель, описывающая в формальном виде атрибуты таких уязвимостей; также приводятся варианты операций над записанной подобным образом моделью.

Статья [16] посвящена визуализации уязвимостей в программном коде. Для этого каждая уязвимость представляется точкой в *N*-мерном пространстве ее признаков, а затем применяется метод главных компонент (из области машинного обучения), понижающий размерность пространства до «понятного» человеку (то есть до двух- или трехмерного). Таким образом, уязвимости представляют собой некоторые сгруппированные «облака» точек, экспертное деление которых на кластеры может означать их классификацию.

Китайские авторы в статье [17] решают задачу автоматической классификации уязвимостей по их описанию, опираясь на статистическую меру TF-IDF (*аббр. от* Term Frequency – Inverse Document Frequency, *пер.* частотность терминов – обратная частотность документов) и глубокие нейронные сети.

Похожий подход рассмотрен японскими исследователями в [18], но с применением таких классификаторов, как метод опорных векторов, логистическая регрессия, дерево решений, метод повышения градиента и случайный лес.

Машинное обучение в интересах автоматической классификации уязвимостей встречается и в других, более ранних работах, например в [19].

В исследовании [20] задача классификации уязвимостей рассматривается как часть общего их прогнозирования по историческим данным. Для этого, в частности, применяется метод k-ближайших соседей (*от англ.* k-nearest neighbors algorithm, k-NN) в приложении к уязвимостям из правительственного хранилища данных NVD (*аббр. от англ.* National Vulnerability Database).

Доклад [21] коллектива ученых и специалистов Рочестерского Института технологий (США) посвящен выявлению характеристик VDO (*сокр. от англ.* Vulnerability Description Ontology) по текстовому описанию уязвимостей, внесенных в международную базу данных CVE (*аббр. от англ.* Common Vulnerabilities and Exposures). Используются следующие классификаторы из области машинного обучения: наивный Байес, дерево решений, случайный лес, метод опорных векторов и AdaBoost (*сокр. от англ.* Adaptive Boosting – адаптивное усиление, алгоритм машинного обучения) на его основе, а также голосование большинством.

Результаты категориального (ориентация – универсальность vs специфичность, и реализация – человек vs автомат) анализа релевантных работ представлены в таблице.

Произведем анализ релевантных работ и их характеристик, представленных в табл. 1. Согласно хронологии публикаций, задача классификации является не новой (в 2013 г. число публикаций на одну больше, чем в 2018 и 2019 гг.) и имеет определенную тенденцию к актуализации (в 2020 г. отмечается двойной всплеск интереса к задаче).

Особенности категоризации классификаций говорят о следующей статистике по всем 20 публикациям: при 18 универсальных классификаций имеется лишь две специальных, а при девяти человеко-ориентированных реализаций имеется 11 автомат-ориентированных и одна публикация относится к обеим ориентациям; при этом наблюдается небольшая тенденция в сторону автоматизации решения задачи классификации. Таким образом, большинство классификаций не учитывают специфику области, в которой применяются; востребованность же в полной автоматизации хотя и незначительно повышается, но полностью переходить на нее, по мнению научной общественности, пока еще рано.

Несмотря на достаточное разнообразие в методах решения задачи, большинство из них полагается на применение машинного обучения (также неразрывно связанного и с автоматизацией решений), что может считаться проверенной и доказавшей свою состоятельность на практике базой.

Таким образом, в интересах разрешения основного противоречия постановки решаемой задачи, полноценные, завершенные и методологически обоснованные способы классификации уязвимостей отсутствуют. Следовательно, требуется разработка способа, суммирующего преимущества и нивелирующего недостатки рассмотренных результатов существующих научных исследований.

Предпосылки к созданию и описание способа классификации

Основная идея вариативного способа классификации уязвимостей в программном коде авторами формулируется следующим образом: *способ классификации уязвимостей должен полагаться как на общие основы безопасности программного кода, так и учитывать его специфику, повышая при этом автоматизацию экспертного труда применением машинного обучения.*

В такой формулировке изначально заложено минимум два, на первый взгляд, антагонистических противоречия (общее vs частное и человек vs автомат), однако именно их разрешение (или, по крайней мере, гармонизация) и позволит создать высокоэффективный, истинно вариативный способ.

Для воплощения этой идеи и построения вариативного способа классификации, которая обладала бы и универсальностью и специфичностью, и «человечностью» и «машинностью» (и все это контекстно и результативно), были использованы следующие предпосылки.

Во-первых, большой объем программного кода, а также огромное число уязвимостей (следуемое, скорее всего, из-за их разнородности) приводит к необходимости максимально возможной автоматизации процесса классификации.

Во-вторых, исходя из сложности формализации уязвимостей, отказаться от экспертного мнения вряд ли удастся, что должно быть определенным образом учтено в процедуре классификации.

В-третьих, сложность применения труда экспертов и субъективность получаемых результатов требует применения машинного обучения, что будет являться существенным плюсом для классификации уязвимостей программного обеспечения.

В-четвертых, для корректности последующей работы с уязвимостями получаемая классификация должна удовлетворять условиям необходимости и достаточности, то есть делить все уязвимости на классы так, чтобы каждая уязвимость однозначно бы относилась к одному классу, притом такой класс всегда бы существовал.

И, в-пятых, для широкого применения способа он должен работать с человеко-ориентированными данными, которыми может являться текстовое описание уязвимостей.

Используя изложенные предпосылки, а также опыт авторов в области исследования феномена «уязвимости программного кода», синтезируем следующие шаги способа их классификации. «Пошаговость» способа обуславливается невозможностью реализации предпосылок неким универсальным приемом.

Шаг 1. Ввести экспертное «стратифицированное» деление классов, то есть выделение нескольких страт. Тем самым, все уязвимости будут относиться к одной из страт (число которых обозначим за S); данное деление является максимально субъективными (эмпирическим, эвристичным).

Шаг 2. Ввести экспертное категориальное деление уязвимостей на группы (данный аппарат хорошо себя зарекомендовал у авторов [22]). Тем самым, каждая уязвимость будет относиться к 2^p. Данное деление является в меньшей степени субъективным, поскольку от эксперта требуется лишь выбрать категориальные пары, притом, что их число ограничено метафизикой предметной области.

Шаг 3. Составить набор классов в виде пересечения введенных делений, получив тем самым $C = S \times 2^P$ классов.

Шаг 4. Собрать базу (или создать ее генератор) уязвимостей для последующего отнесения их к каждому классу. Тем самым, будет получено N уязвимостей, которые будут относиться к C классам.

Шаг 5. Составить текстовое (человеко-ориентированное) описание уязвимостей из созданной базы. Для этого можно привлечь группу специалистов в области информационной безопасности кода, не обязательно являющихся экспертами. Тем самым, будет получено N описаний уязвимостей.

Шаг 6. Выделить признаки в текстовом описании уязвимостей (например, «мешок слов» или TF-IDF). Таким образом, для всех N описаний уязвимостей будет получено по F признаков.

Шаг 7. Ввести для каждой уязвимости уникальный идентификатор, хранящий в себе ее особенности (например, в виде поля бит), что позволит как определить математические операции с уязвимостями, так и более точно относить их к классам.

Шаг 8. Создать модель машинного обучения для классификации уязвимостей. Для этого будут использоваться входные данные (обучающая выборка) в виде *N* групп по *F* признаков, соответствующие одному из *C* классов.

Учет специфики программного кода может осуществляться экспертом как на Шаге 1, так и на Шаге 2, тем самым гармонизируя *первое* противоречие идеи. Такая специфика может заключаться в монолитности и слабой структурированности операций (для машинного кода), «перемешивании» различной метаинформации (для мультипарадигмального исходного кода), применении графических элементов (для блок-схем) и т.п.

Повышение степени автоматизации способа осуществляется постепенно, пошагово, в зависимости от эвристичности методов их реализации: от чисто «ручных» (Шаг 1 и Шаг 2), до полностью «машинных» (Шаг 7 и Шаг 8). Срединная часть способа допускает вариативность реализации (за исключением Шага 5, который на данном этапе развития человеческой мысли пока не может быть полностью реализован автоматом), что гармонизирует *второе* противоречие идеи.

Пример

Для обоснования применимости предложенного способа в качестве примера произведем с его помощью классификацию уязвимостей телекоммуникационных устройств. Программный код из области телекоммуникаций обладает следующей спецификой: работает во встроенных устройствах (маршрутизаторах и т.п.), не ориентирован на работу с пользователем (поскольку имеет ограниченные механизмы пользовательского ввода/вывода в виде удаленной консоли, Web-интерфейса), работает с сетью, обрабатывает огромные потоки информации, нарушение его доступности напрямую влияет на функционирование всей сети, не имеет множества работающих с устройством пользователей, не выполняет пользовательские приложения, атакуется в основном по сетевым векторам и др.

Шаг 1. Применение стратификации к уязвимостям программного кода телекоммуникационных устройств может опираться на предыдущие исследования авторов и построенные ими модели машинного кода [23]. Тогда уязвимости разного структурного уровня в коде могут инспирировать аналогичные страты таких же уровней (от младшей, наиболее близкой к конкретной функциональности кода, к старшей, повышающей уровень абстракции): атомарный (А), низкоуровневый (Н), среднеуровневый (С), высокоуровневый (В), концептуальный (К). Подчеркнем, что уязвимости всех страт, кроме первой, в основном появляются в результате действий разработчиков программного обеспечения, а атомарная – это, как правило, следствие ошибок в работе средств сборки кода (компилятора, ассемблера, линкера и пр.). Очевидно, что в данном делении не учитывается специфика программного обеспечения телекоммуникационных устройств, и, следовательно, она должна проявиться на следующем шаге.

Шаг 2. На данном шаге необходимо применить категориальное деление множества уязвимостей на группы, при этом учитывая специфику телекоммуникационных

Конкретное потому конкретно, что оно есть синтез многих определений, следовательно, единство многообразного.
Карл Маркс

устройств. Для этого требуется выбрать соответствующие философские категории пар-антагонистов, которые могут быть следующими: Анализ vs Синтез и Основной vs Вспомогательный. Выбор первой пары обусловлен тем, что функционирование любого программно-аппаратного устройства обработки потока информации (к которому относятся практически все телекоммуникационные устройства) построено по единому принципу ее преобразования: получить и проанализировать входные данные, а затем синтезировать и выдать выходные данные. При этом центральный этап обработки работает по такому же принципу, просто с использованием более атомарных правил преобразования. Таким образом, часть программного кода в таких устройствах занимается анализом, а часть – синтезом. Выбор второй категориальной пары обусловлен близостью кода к основному назначению, поскольку функциональность телекоммуникационных устройств «крутится» вокруг обработки полей пакетов для различных уровней OSI (Open Systems Interconnection – сетевая модель взаимодействия открытых систем), дополняясь вспомогательными возможностями (например, построением виртуальных сетей и поддержкой конфигурационных файлов).

Исходя из предложенных двух категориальных пар, комбинация их элементов дает четыре деления уязвимостей – ОА, ВА, ОС и ВС – по их нахождению в программном коде с некоторой функциональностью. Укажем интерпретацию каждой (и, следовательно, ее уязвимости):

– ОА (Основной Анализ) - осуществляющий разбор и обработку всех поступающих сетевых пакетов, расшифровку, сборку в сессии и пр.;

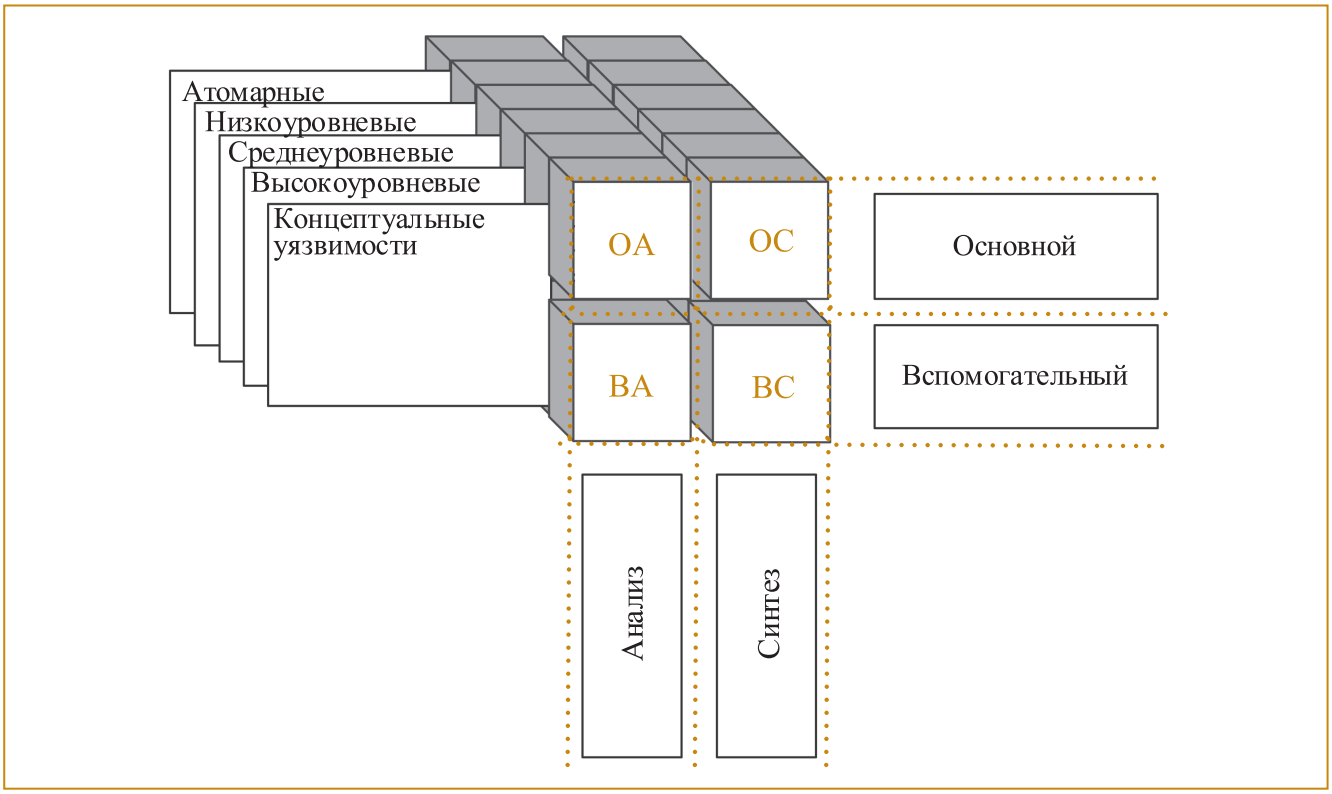
– ВА (Вспомогательный Анализ) - дополняющий обработку сетевых пакетов вспомогательными функциями, такими как проверка корректности пакетов, сравнение идентификаторов из хранилища данных и пр.;

– ОС (Основной Синтез) - осуществляющий генерацию исходящих сетевых пакетов, их шифрование, выбор интерфейса для отправки и пр.;

– ВС (Вспомогательный Синтез) - дополняющий создание сетевых пакетов вспомогательными функциями, такими как изменение параметра TTL (Time To Live – время «жизни» пакетов), обновление внутреннего лога, генерация уникальных идентификаторов и ключей и пр.

Шаг 3. Пять страт, полученные на Шаге 1, а также четыре комбинации элементов категориальных пар, полученные на Шаге 2, в совокупности позволяют создать стратифицированную иерархию уязвимостей, состоящую из 5 x 4 = 20 классов; графическая схема процесса их получения представлена на рисунке.

Таким образом, все «сущностное» множество программного кода телекоммуникационных устройств в 3D-пространстве (в виде совокупности функциональных



Графическая схема получения классификаций уязвимостей

действий – полученной категориальным делением по 2D-плоскости, и их реализаций на различных структурных уровнях – полученных делением на страты по 1D-оси) разбивается на набор из 20 «кубиков» такой же 3D-размерности, как и само «пространство». То есть полученные подобным образом классы совмещают две сущности – функциональную (после Шага 2) и реализационную (после Шага 1), что существующие классификации, как правило, не позволяют.

Шаг 4. Суть шага заключается в отнесении уязвимостей из некоторой базы к классам, полученным на предыдущем шаге, что должно быть использовано на заключительном шаге способа при построении модели машинного обучения. Поскольку это требуется лишь для полноценной рабочей классификации, а назначением примера является подтверждение работоспособности способа, то достаточно будет привести «образчики» уязвимостей телекоммуникационных устройств (имеющих встроенное программное обеспечение в виде сетевой ОС, построенной на базе модулей, выполняющих основные и вспомогательные функции по обработке сетевого трафика) для классов, показав тем самым обоснованность и адекватность последних; уязвимости (У) будем обозначать, как $У_{XZ}$, где X – первая буква названия страты (т.е. $X \in$ {А, Н, С, В, К}, а Z – аббревиатура категориальной пары (т.е. $Z \in$ {ОА, ВА, ОС, ВС}).

Рассмотрим возможные прототипы уязвимости для каждого из 20 классов.

1) УАОА – из-за ошибочного использования для хранение контрольной суммы из поля TCP-заголовка однобайтового регистра вместо двухбайтового данный механизм проверки может работать некорректно. Подобного рода атомарные ошибки (низшего уровня абстракции) могут появиться именно в момент компиляции исходного кода (а не его написания, подверженного многократным проверкам разработчиков), поскольку любой современный компилятор является достаточно сложным программным средством, обладающим нетривиальными алгоритмами работы (например, на фазах оптимизации внутреннего представления программы или генерации по нему выполняемого кода), тестирование которых с полным покрытием всех случаев является крайне сложной задачей [24].

2) $У_{НОА}$ – из-за ошибки переполнения буфера модуль разбора сетевых пакетов может выполнить команду злоумышленника.

3) $У_{СОА}$ – из-за ошибки в условии проверки диапазона IP-адресов модуль проверки входных сетевых пакетов может его ошибочно отбросить.

4) $У_{ВОА}$ – из-за неверной реализации внутренней политики безопасности простой пользователь системы может получить доступ к настройкам, доступным только администратору.

5) $У_{КОА}$ – из-за отсутствия защиты встроенной системы отладки (например, в виде открытого порта для доступа разработчиков к настройкам системы) злоумышленник может получить полный доступ к устройству.

6) $У_{АОС}$ – из-за ошибочного использования инструкций процессора для работы с индексом массива сетевых пакетов как со знаковым вместо беззнакового часть пакетов может пропасть, а часть – быть заменена на «мусорные», расположенные в памяти перед началом массива.

7) $У_{НОС}$ – из-за ошибки переполнения буфера модуль генерации сетевых пакетов может «испортить» байтовую последовательность других пакетов.

8) $У_{СОС}$ – из-за ошибки в ходе проверки введенных данных при аутентификации, ведущей к возможности несанкционированного доступа в систему, что аналогично уязвимости CVE-2021-27215, имеющей следующее описание – «Web-интерфейс продукта Genugate[2] некоторых версий использует различные методы аутентификации пользователей, один из которых не проверяет предоставляемые данные, что позволяет злоумышленнику войти в панель администратора от имени любого пользователя с необходимыми правами».

9) $У_{ВОС}$ – из-за выбора недостаточной длины ключа шифрования возможно нарушение конфиденциальности передаваемых данных (за разумное время) [25]. При этом подобного рода критичная ошибка может быть допущена сознательно для ослабления уровня защиты передаваемого сетевого трафика с целью его дальнейшей расшифровки. Так, компания Crypto AG, принадлежащая ЦРУ и специализирующаяся на коммуникациях и информационной безопасности, ослабляла шифрование в своих продуктах с целью шпионажа путем взлома передаваемых данных (https://www.bbc.co.uk/programmes/m000w499).

10) $У_{КОС}$ – из-за выбора слабого алгоритма шифрования возможно нарушение конфиденциальности передаваемых данных (за разумное время).

11) $У_{АВА}$ – из-за ошибочного использования операций сдвига бит числа может быть неверно вычислен диапазон IP-адресов сети, что приведет к общему нарушению маршрутизации сетевых пакетов.

12) $У_{НВА}$ – из-за ошибки переполнения буфера модуль разбора командной строки злоумышленник, подключившись к консоли, может повысить себе права. Подобная ошибка, но с итоговым сбоем программы, может быть при целочисленном переполнении с последующим доступом к элементу массива меньшего размера, как в случае уязвимости CVE-2021-30022 (имеющей следующее описание – «Целочисленное переполнение в продукте GPAC[3] некоторой версии приводит к тому, что переменная (pps_id) принимает отрицательное значение и проходит проверку на превышение размера массива, а последующее ее использование в качестве индекса массива (avc->pps) делает попытку доступа к неверному адресу памяти, что ведет к сбою программы»).

13) $У_{СВА}$ – из-за отсутствия проверки деления на «0» модуль анализа конфигурационных файлов может вызвать исключение доступа к памяти и нарушить работу всего устройства.

14) $У_{ВВА}$ – из-за наличия учетных записей «по умолчанию», ошибочно оставленных в готовом продукте после его отладки, злоумышленник может получить несанкционированный доступ к системе.

15) $У_{КВА}$ – из-за непродуманной модели управления ресурсами (доступная оперативная память, процессорное время, файловая система) модулям может не хватать ресурсов для обработки и хранения поступающих сетевых пакетов, что будет приводить к их потере.

16) $У_{АВС}$ – из-за ошибочного использования арифметических операций для работы с беззнаковыми переменными вместо знаковых может быть неверно вычислено время следующего программного события, что приведет к общему нарушению функционирования.

17) $У_{НВС}$ – из-за ошибки переполнения буфера модуль генерации массива с псевдослучайными числами может вызвать исключение доступа к памяти и нарушить работу всего устройства.

18) $У_{СВС}$ – аналогично п. 13.

19) $У_{ВВС}$ – из-за выбора слабого механизма генерации псевдослучайных чисел существует вероятность их предсказания и нарушение конфиденциальности передаваемых данных.

20) $У_{КВС}$ – аналогично п. 15.

Таким образом, для каждого из приведенных классов существует хотя бы одна реалистичная (имеющая вероятность появления, отличную от нуля) уязвимость, что говорит об адекватности полученного деления.

Список литературы

1. *Розова С.С.* Классификационная проблема в современной науке. Новосибирск: Наука, 1986. 224 с.
2. *Виткова Л.А., Израилов К.Е., Чечулин А.А.* Классификация уязвимостей интерфейсов транспортной инфраструктуры умного города // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО-2020). Тр. IX междунар. научно-технич. и методич. конф. С.-Петербург. 2020. С. 253-258.
3. *Понкин И.В., Редькина А.И.* Классификация как метод научного исследования, в частности в юридической науке // Вестник Пермского университета. Юридические науки. 2017. Вып. 37. С. 249-259.
4. *Джангазова К.А., Утамуратов О.А.* Классификация уязвимостей безопасности // Высшая школа. 2016. № 10. С. 123-124.
5. *Надеждин Е.Н., Шершакова Т.Л.* Задача классификации уязвимостей программного обеспечения в корпоративной информационной сети // Шуйская сессия студентов, аспирантов, педагогов, молодых ученых. Тр. X междунар. научной конф. 2017. С. 207.
6. *Надеждин Е.Н., Шершакова Т.Л.* Алгоритм нечеткой классификации уязвимостей прикладного программного обеспечения // Проблемы фундаментальной и прикладной информатики в управлении, автоматизации и мехатронике. Тр. междунар. научно-технич. конф. Курск. 2017. С. 109-113.

[2] Межсетевой экран Genugate производства Genua представляет собой гибридное программно-аппаратное решение, объединяющее функции пакетного фильтра и шлюза уровня приложений. Источник: https://www.anti-malware.ru/news/2021-03-01-114534/35142

[3] Кроссплатформенный фреймворк для работы с мультимедиа-файлами, в частности, с презентациями, графикой, интерактивностью и анимацией. Поддерживается большинством известных операционных систем, в частности Windows, Linux, MacOSX, Android, iOS и т.д. Источник: https://besplatnye-programmy.com/raznoe-dlya-razrabotchikov/2501-gpac.html

7. *Матяш Е.Д.* Классификация и разметка элементов текста на уровне предложения для выявления уязвимостей нулевого дня // Комплексная защита информации. Тр. XXII научно-практич. конф. Полоцк. 2017. С. 253-256
8. *Доронин А.К., Липницкий В.А.* Построение модели машинного обучения для задачи классификации степени критичности CVE-уязвимостей // Веснік Магілеўскага дзяржаўнага ўніверсітэта імя А.А. Куляшова. Серыя В. Прыродазнаўчыя навукі: матэматыка, фізіка, біялогія. 2020. № 1 (55). С. 51-63.
9. *Алексеев И.В., Зегжда П.Д.* Классификация уязвимостей сетевых протоколов на основе спецификаций // Проблемы информационной безопасности. Компьютерные системы. 2020. № 1. С. 24-32.
10. *Зиновьев А.В., Язиков Г.Е., Аборнев А.А.* Практическое применение метода машинного обучения «мешок слов» в модуле «1С (чат-бот)» для автоматизации работы сотрудников первой линии поддержки пользователей // Газовая промышленность. 2021. № 5 (816). С. 28-31.
11. *Израилов К.Е.* Обобщенная классификация уязвимостей интерфейсов транспортной инфраструктуры умного города // Информационные технологии. 2021. Т. 27. № 6. С. 330-336.
12. *Муханова А., Ревнивых А.В., Федотов А.М.* Классификация угроз и уязвимостей информационной безопасности в корпоративных системах // Вестник НГУ. Серия: Информационные технологии. 2013. Т. 11. № 2. С. 55-72.
13. *Сучков А.П.* Классификация уязвимостей интегрированных систем управления на ранних стадиях жизненного цикла // Системы и средства информатики. 2017. Т. 27. № 4. С. 132-143.
14. *Сидоров А.Г.* Способы классификации и системы оценки уязвимостей в аналитических системах сбора и анализа сведений об уязвимостях и угрозах ИБ // Фундаментальные и прикладные исследования молодых ученых: Тр. II междунар. научно-практич. конф. студентов, аспирантов и молодых ученых. Омск. 2018. С. 503-509.
15. *Кубарев А.В.* Подход к формализации уязвимостей информационных систем на основе их классификационных признаков // Вопросы кибербезопасности. 2013. № 2 (2). С. 29-33.
16. *Израилов К.Е.* Визуализация многопризнаковых уязвимостей программного кода с помощью метода главных компонент // Вестник СПбГУПТД. Серия 1: Естественные и технические науки. 2020. № 1. С. 3-8.
17. *Huang G., Li Y., Wang Q., Ren J., Cheng Y., Zhao X.* Automatic Classification Method for Software Vulnerability Based on Deep Neural Network // The Proceedings of IEEE Access. 2019. Vol. 7. PP. 28291-28298.
18. *Aota M., Kanehara H., Kubo M., Murata N., Sun B., Takahashi T.* Automation of Vulnerability Classification from its Description using Machine Learning // The Proceedings of Symposium on Computers and Communications (ISCC). IEEE, 2020. PP. 1-7.
19. *Shuai B., Li H., Li M., Zhang Q., Tang C.* Automatic classification for vulnerability based on machine learning // The Proceedings of International Conference on Information and Automation (ICIA). IEEE, 2013. PP. 312-318.
20. *Last D.* Using historical software vulnerability data to forecast future vulnerabilities // Resilience Week (RWS). 2015. PP. 1-7.
21. *Gonzalez D., H. Hastings and M. Mirakhorli.* Automated Characterization of Software Vulnerabilities // 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME). 2019. PP. 135-139.
22. *Буйневич М.В., Израилов К.Е.* Категориальный синтез и технологический анализ вариантов безопасного импортозамещения программного обеспечения телекоммуникационных устройств // Информационные технологии и телекоммуникации. 2016. Т. 4. № 3. С. 95-106.
23. *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.
24. *Richards T., Walters E.K., Moss J.E.B., Palmer T., Weems C.C.* Towards universal code generator generation» // The Proceedings of IEEE International Symposium on Parallel and Distributed Processing. 2008. PP. 2560-2567.
25. *Волков Д.С., Иванов А.В.* Применение генетических алгоритмов для выбора ключа шифрования в системах связи ракетных комплексов // Актуальные проблемы авиации и космонавтики. 2016. Т. 1. № 12. С. 145-147.

Буйневич Михаил Викторович *– д-р техн. наук, проф., профессор кафедры прикладной математики и информационных технологий Санкт-Петербургского университета государственной противопожарной службы МЧС России,*

Израилов Константин Евгеньевич *– канд. техн. наук, доцент кафедры защищенных систем связи Санкт-Петербургского государственного университета телекоммуникаций им. проф. М.А. Бонч-Бруевича, старший научный сотрудник лаборатории проблем компьютерной безопасности Санкт-Петербургского Федерального исследовательского центра РАН,*

Матвеев Владимир Владимирович *- д-р техн. наук, проф., профессор кафедры экономической безопасности Санкт-Петербургского государственного экономического университета,*

Покусов Виктор Владимирович *- председатель Казахстанской ассоциации информационной безопасности, Алматы, Казахстан.*

E-mail: bmv1958@yandex.ru, konstantin.izrailov@mail.ru, 070355mvv@gmail.com, v@victor.kz