Обзорная статья
УДК 004.056; 004.8
DOI:10.31854/1813-324X-2023-9-6-83-94

Машинное обучение vs поиск уязвимостей в программном обеспечении: анализ применимости и синтез концептуальной системы

Николай Викторович Леонов[1], Leonov-nv@yandex.ru
Михаил Викторович Буйневич[2] ✉, bmv1958@yandex.ru

[1]Государственный научно-исследовательский институт прикладных проблем,
Санкт-Петербург, 191167, Российская Федерация
[2]Санкт-Петербургский университет государственной противопожарной службы МЧС России,
Санкт-Петербург, 196105, Российская Федерация

Аннотация: *Работа посвящена проблеме поиска уязвимостей в программном обеспечении, а также возможностям применения такого перспективного направления информационных технологий, как машинное обучение. Для этого произведен обзор научных публикаций предметной области из российской и зарубежной базы цитирования. Осуществлен сравнительный анализ результатов обзора по следующим критериям: год публикации, область применения, идея, решаемая задача машинного обучения, степень реализации его моделей и методов; по каждому критерию сделаны основополагающие выводы. В итоге предложены 7 принципов построения новой концептуальной системы поиска уязвимостей в программном обеспечении с помощью машинного обучения, краткий смысл которых состоит в следующем: многостороннее исследование программы, комбинирование известных методов, использование машинного обучения в каждом методе и алгоритме управления ими, возможность корректировки работы экспертом, хранение информации в базе данных и ее синхронизации с внешними, рекомендательный характер относительно найденных уязвимостей; использование единой программно-аппаратной платформы. На основании декларируемых принципов разработана графическая схема такой системы.*

Ключевые слова: *информационная безопасность, поиск уязвимостей, машинное обучение, обзор, принципы, система поиска*

Ссылка для цитирования: 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

Machine Learning vs Software Vulnerability Detection: Applicability Analysis and Conceptual System Synthesis

Nikolay Leonov[1], Leonov-nv@yandex.ru
Mikhail Buinevich[2] ✉, bmv1958@yandex.ru

[1]State Research Institute of Applied Problems,
St. Petersburg, 191167, Russian Federation
[2]Saint-Petersburg University of State Fire Service of EMERCOM of Russia,
St. Petersburg, 196105, Russian Federation

Abstract: *The article is devoted to the searching for vulnerabilities in software problem, as well as the possibilities of application of such a promising area in information technology as machine learning. For this purpose, a review of scientific publications in this area from Russian and foreign citation databases is made. A comparative analysis of the review's results was made according to the following criteria: publication year, application field, idea, solved problem of machine learning, degree of realization of its models and methods; for each criterion basic conclusions were drawn. As a result, 7 principles of building a new conceptual system of searching for vulnerabilities in software with the help of machine learning are proposed, the short meaning of which is as follows: program's multilateral study, combination of known methods, the use of machine learning in each method and algorithm of its management, the possibility of correcting the expert's work, storing information in a database and its synchronization with external, advisory nature of the found vulnerabilities; single software application usage. Based on the stated principles, a graphical scheme of such a system has been developed.*

Keywords: *information security, vulnerability search, machine learning, review, principles, search engine*

For citation: Leonov N., Buinevich M. Machine Learning vs Software Vulnerability Detection: Applicability Analysis and Conceptual System Synthesis. *Proceedings of Telecommun. Univ.* 2023;9(6):83–94. DOI:10.31854/1813-324X-2023-9-6-83-94

Введение

Безопасность программного обеспечения (ПО) является актуальнейшей проблемой современного мира, практически полностью функционирующего на базе и законах информационных технологий. Одним из наиболее эффективных способов повышения безопасности ПО (хотя и не единственным) небезосновательно признан поиск уязвимостей в программах с последующей их нейтрализацией или попросту отказом от дальнейшего использования. Наличие уязвимостей в программах может приводить к критическим последствиям в любой области – начиная от бытовых «умных» устройств или персональных компьютеров и заканчивая обороноспособностью целой страны. Постоянно растущее количество такого рода информационных угроз позволяет утверждать, что данный способ (имеется в виду – поиск уязвимостей) достиг некоторого «научно-технического тупика», для выхода из которого требуется применение качественно новых подходов.

В качестве одного из них гипотетически может выступить машинное обучение (МО), уже доказавшее свою применимость для решения огромного спектра разнородных прикладных задач: программный инжиниринг [1], управление состоянием дорожного покрытия [2], построение сетей 5G [3], повышение обучаемости студентов [4], обеспечение непрерывности функционирования Интернета вещей [5], – и многие другие. Исходя из этих соображений, далее в статье будет проведен аналитический обзор возможностей МО, которые могут быть применены для поиска уязвимостей в ПО – что и определяет предметную область настоящего исследования.

Обзор релевантных работ

Произведем обзор научных публикаций отечественных и зарубежных ученых, посвященных задаче поиска уязвимостей в ПО, для решения которой применяются модели и методы МО; поиск осуществлялся по базам РИНЦ и IEEE Xplorer, соответственно, а в качестве конкретных статей и результатов интеллектуальной деятельности отбирались наиболее релевантные запросу «машинное обучение, поиск уязвимостей». В каждом обзоре, по возможности, будут отражены следующие аспекты исследований: область применения (**Область**), метод поиска (**Метод**), решаемая МО задача (**Задача**), реализация МО (**Реализация**) и степень готовности решения (**Степень**).

Работа [6] посвящена безопасности Web-приложений (**Область**), в интересах чего авторы описывают разработанный сканер уязвимостей – BTB (*аббр. от англ.* Bug Terminating Bot, *перев. на русс.* – бот для устранения ошибок). Суть работы BTB заключается в сканировании страниц Web-приложений и тестировании их работоспособности под «полезной» нагрузкой (например, осуществляя SQL-инъекцию путем добавления специальных символов в конец запросов); возникновение нештатных ситуаций, таких, как ошибка выполнения, говорят и о наличии уязвимости (**Метод**). Также в BTB присутствует компонент исправления уязвимостей, предлагая необходимый для этого код. BTB имеет вид прототипа, разработанного на языке C# и использующего SQL-сервер в качестве собственной базы данных (БД). Применяемое авторами МО построено на базе SVM (**Реализация**) (*аббр. от англ.* Support Vector Machine, *перев. на русс*. машины опорных векторов) и используется на специальном сервере продукта (**Степень**) для повышения эффективности сканера, предлагая ему улучшенные параметры работы. Задачами МО, решаемыми в BTB, являются классификация и регрессионный анализ (**Задача**).

В аналитическом исследовании [7] оценивается текущее состояние поиска программных уязвимостей (**Область**) с применением МО, а также делаются прогнозы на будущее. В работе выделяются различные представления и векторизации кода программ (например, текстового ассемблерного,

полученного из бинарного машинного), состоящие из следующих: метрики программного инжиниринга, языковые модели описания кода, синтаксическое дерево (еще называемое деревом абстрактного синтаксиса, *перев. на англ*. Abstract Syntax Tree, *аббр*. AST) с семантической информацией, различные графы (потока управления, межпрограммных зависимостей и т. п.).

В качестве методов МО рассматриваются 2 типа-антагониста: контролируемые (в случае наличия пометок об уязвимых местах) и неконтролируемые (когда информация об уязвимостях изначально отсутствует) (**Метод**). В качестве реализации методов первого типа приводятся примеры применения Наивного Байеса, SVM, случайного леса, CNN (*аббр. от англ*. Convolutional Neural Networks, *перев. на русс.* сверточная нейронная сеть, модели последовательности, BLSTM (*аббр. от англ*. Bidirectional Long Short-Term Memory, *перев. на русс.* нейронные сети с двунаправленной долговременной краткосрочной памятью) и др.; указывается, что решения из двух последних примеров превосходят остальные (**Реализация**). В основу методов второго типа заложен поиск уязвимостей по шаблонам (**Метод**), созданным экспертным способом. Для этого в части МО применяется понижение размерности, кластеризация API-вызовов, выявление аномалий, а также анализ искажений (**Задача**) (например, оценка влияния переменных, полученных из испорченных пользовательских, на поведение системы).

В выводах статьи подчеркивается высокий потенциал методов поиска уязвимостей на базе МО; также приводятся новые возможные подходы (**Степень**), основанные на символьном выполнении, скрытых моделях Маркова и RNN (*аббр. от англ*. Recurrent Neural Network, рекуррентные нейронные сети).

Работа [8] посвящена разработке сканеров выявления SQL-инъекций (**Метод**) для Web-сайтов (**Область**). Для реализации МО в части классификации вредоносных SQL-команд (**Задача**) протестированы модели Наивного Байеса, SVM, дерево решений, случайный лес (**Реализация**), а также библиотеки XGBoost и CatBoost. Обучение производилось на наборах данных, содержащих валидные и небезопасные команды. Указывается, что предварительные эксперименты говорят о значении F-меры около 0.95, что, безусловно, является высоким показателем (**Степень**).

В статье [9] решается задача выявления уязвимостей типа SQL-инъекций, содержащихся в коде языка программирования PHP (**Область**). В интересах применения МО автор с помощью собственного скрипта (**Степень**) выявляет некоторые признаки исследуемого кода (**Метод**), содержащие наличие проверок и очисток ввода (актуальных для противодействия данной уязвимости) – как стандартные, так и оригинальные. Для обработки признаков при подачи их формализованного представления на вход модели МО применялись две техники – 1) «мешок слов», позволяющий избавляться от терминов в коде, не связанных с SQL-инъекциями; 2) Word2Vec, учитывающая контекст функций [10]. Для определения наиболее эффективной реализации МО автор проводит эксперименты, применяя различные классификаторы (**Задача**): для 1-й техники – дерево решений, случайный лес, SVM, логистическая регрессия, MLP (*аббр. от англ*. Multi-Layer Perceptron, *перев. на русс*. многоуровневый перцептрон или персептрон), RNN, BLSTM и CNN; для 2-й техники – последние 4 классификатора из первой техники (**Реализация**). Исходя из важности для настоящего исследования полученных результатов, они продублированы в таблице 1.

ТАБЛИЦА 1. Результаты экспериментов по классификации SQL-инъекций

TABLE 1. Experimental Results of SQL Injection Categorization

Классификатор	Полнота	Точность	F-мера
для «мешка слов»			
Дерево решений	56.5 %	93.4 %	0.650
Случайный лес	57.7 %	93.6 %	0.660
SVM	58.3 %	95.4 %	0.732
Логистическая регрессия	56.0 %	95.1 %	0.713
MLP	63.7 %	95.3 %	0.746
RNN	62.4 %	95.3 %	0.742
BLSTM	61.4 %	95.2 %	0.734
CNN	59.9 %	95.3 %	0.734
для Word2Vec			
MLP	50.7 %	92.8 %	0.601
RNN	49.9 %	92.7 %	0.595
BLSTM	48.5 %	93.2 %	0.606
CNN	48.1 %	92.5 %	0.573

Согласно полученным результатам (см. таблицу 1), делаются следующие выводы. Во-первых, F-мера определения уязвимостей оказалась выше при использовании «мешка слов». Во-вторых, наилучшего результата в случае «мешка слов» достиг MLP, а в случае Word2Vec – BLSTM. В-третьих, работа классификаторов глубокого обучения (т. е. последних четырех) для каждой техники оказалась лучше классических. И, в-четвертых, необходимо отметить низкое значение полноты (менее 66 %), что негативно влияет на количество выявленных уязвимостей.

В эмпирическом исследовании [11] указывается, что модели обнаружения уязвимостей в программном коде (**Область**) на основе глубокого обучения достигли значительного прогресса, пре-

вышающего в ряде случаев даже инструменты статического анализа. Тем не менее, понимание возможностей таких моделей еще далеко от конечной точки; в связи с чем в данной работе делаются экспериментальные оценки границ их применимости (**Степень**). В частности, авторы стремятся ответить на следующие вопросы:

1) Согласуются ли модели с реально обнаруженными уязвимостями?

2) Как на результативность обнаружения влияет тип уязвимости?

3) Сложнее ли делать предсказания для программ по определенным признакам кода?

4) Влияет ли увеличение датасета на эффективность поиска?

5) Как состав обучающих данных влияет на эту эффективность?

6) Какая информация об исходном коде использовалась моделями для предсказания уязвимостей?

Для ответа на вопросы авторы в своей работе воспроизвели 11 моделей глубокого обучения, построенных на абстрактных архитектурах (GNN, RNN, BLSTM, CNN и «Transformer» или преобразователь данных (**Реализация**) и обучаемых на конкретных датасетах. Модели применялись для классификации экземпляров кода на уязвимые и безопасные (**Задача**), используя для этого его различные признаки (по AST, графам потоков управления, зависимостям данных и др.) (**Метод**). Ниже все эти модели перечислены в нотации «название / архитектура / датасет(ы)»:

– Devign / GNN (*аббр. от англ*. Graph Neural Network, *перев. на русс*. графовая нейронная сеть / Devign;

– ReVeal / GNN, граф свойств / Devign, ReVeal;

– ReGVD / GNN, токен / Devign;

– CodeBERT / преобразователь данных / Devign;

– VulBERTa-CNN / преобразователь данных, CNN / VulDeePecker, Draper, ReVeal, μVulDeePecker, Devign, D2A;

– VulBERTa-MLP /преобразователь данных / MLP;

– PLBART / преобразователь данных / Devign;

– LineVul / преобразователь данных / MSR (*аббр. от англ.* Multi-Sentence Resampling);

– Code2Vec / MLP, AST / Devign;

– SeSyVR / RNN / SARD (*аббр. от англ*. Software Assurance Reference Dataset), NVD (*аббр. от англ*. National Vulnerability Database);

– VulDeeLocator / BLSTM / SARD, NVD.

Были получены следующие ответы на соответствующие вопросы, поставленные исследователями:

1) 34,9 % тестовых данных (или 30,6 % общих данных) дают разные прогнозы в зависимости от используемых при обучении моделей.

2) Все модели с разной эффективностью способны предсказывать разные типы уязвимостей (в экспериментах учитывались следующие из них: переполнение буфера, повышение привилегий, а также ошибки значений, ресурсов и проверки ввода); так, например, Devign и ReVeal предсказывали все типы уязвимостей с близкой вероятностью, для других же решений ошибки ресурсов обнаруживались хуже всего.

3) Предсказания всех моделей оказались лучше для простого набора данных (т. е. признаков кода), чем для сложного, с разницей в 10 %.

4) Увеличение датасета улучшает эффективность предсказания уязвимостей, но незначительно – при изменении набора данных от 10 % до 100 % от исходного, F-мера увеличилась на 0.16.

5) Разнообразие обучающих данных из разных проектов не дает преимуществ по сравнению с использованием данных только из одного проекта (для примера был взят набор Chrome из датасета MSR, содержащий 76 тысяч образцов).

6) Авторы получили матрицу схожести важных признаков для парного применения моделей (используя простое пересечение и меру Жаккара); так, хотя предсказания моделей имеют различную эффективность, однако используемая ими информация о коде пересекается.

Одной из основных предпосылок исследования [12] является автоматизация поиска уязвимостей в различных средах: сетях, Web-сайтах, программном обеспечении. В качестве обоснования к применению МО указаны сложность типового исследуемого объекта и невозможность создания полной базы всех существующих уязвимостей. Процесс поиска уязвимостей авторами описывается с помощью модели на базе марковского процесса принятия решения (**Метод**), когда под состоянием системы понимаются сведения об объекте исследования, а применение утилиты – как оптимизация действий по сбору дополнительных данных (**Задача**). В интересах этого предлагается применять соответствующую модель МО с подкреплением; в частности, на базе алгоритма Q-обучения (**Реализация**), характерного для агентного подхода. Описанные предложения были оформлены в виде программного прототипа (**Степень**) модели поиска уязвимостей Web-ресурсов (**Область**). Действиями по сбору информации являлись основные функции сканеров Web-сайтов – nmap, SQLmap, WhatWeb и XSScrapy. Состояние системы описывалось с помощью 128 элементов, включающих версию Web-сервера, его поддомены, XSS уязвимости и др. Используя датасет различных характеристик сайтов, модель была успешно обучена за 60 эпох – полных итераций по набору данных в процессе обучения. Также, авторы подчеркивают незаконченность предложенного решения, поскольку для реального его применения количество характеристик должно быть, по их мнению, примерно в 100 раз больше.

В другой аналитической работе [13], проведенной одним из соавторов настоящей статьи, осуществлена систематизация основных этапов статического анализа кода программ (**Область**) в интересах поиска уязвимостей с позиции применимости решения различных задач МО (**Метод**). Так, выделены следующие этапы поиска: сбор данных, подготовка объекта, проведение эксперимента и формирование результатов. С другой стороны, рассматриваются следующие задачи МО: классификация, выявление аномалий, регрессия, кластеризация и обобщение (**Задача**). Для более половины комбинаций этапов и задач приведены примеры обоснованности решений в интересах поиска. Так, например, классификация частей кода на легальные и небезопасные позволит обнаруживать существующие и близкие к ним уязвимости; а кластеризация, отнесение последних к отдельным классам. В результате была предложена обобщенная, с методологических позиций, модель (**Степень**), систематизирующая решения частных задач статического анализа в выбранной авторами статьи предметной области с применением МО. Какие-либо конкретные предложения по реализации решения задач МО в статье отсутствуют (**Реализация**).

В [14] поиск уязвимости в программах, имеющих форму машинного кода (**Область**), предлагается осуществлять с применением глубокой RNN для BLSTM (**Реализация**). При этом уязвимость интерпретируется, как некоторые абстрактные особенности кода – семантики инструкций, их влияния на регистры и др. Само МО применяется для решения типовой задачи классификации (**Задача**) исходных образцов программ на безопасные или содержащие уязвимости. Метод построения классификатора состоит из следующих шагов (**Метод**): 1) преобразование инструкций машинного кода в векторное представление, используя обучение с переносом [15], что позволяет делать семантически близкие инструкции, имеющими подобные представления; 2) построение RNN, входными признаками которой являются векторные представления инструкций; 3) фиксирование модели, готовой для применения. Авторы заявляют о выявлении уязвимости типа целочисленного переполнения с 84 %-ной точностью (**Степень**).

Направлением исследования в [16] является противодействие уязвимостям в локальных сетях и Web-приложениях (**Область**). Изначально указывается, что сфера информационных технологий стала одной из первых, в которые стали внедряться технологии искусственного интеллекта (ИИ); одной из причин этого авторы называют огромный объем данных, «неподъемный» для человека. Упоминаются некоторые системы автоматического поиска уязвимостей на базе ИИ: Nessus и OpenVAS – для локальных сетей, SQL-map и what-web – для Web-ресурсов, антивирус Kaspersky – для ПО. Подчеркивается, что само по себе внедрение такого рода интеллектуальных систем может служить причиной появления новых уязвимостей из-за их собственной небезопасности; также, существуют и атаки на сами системы. Отмечаются успехи, достигнутые с помощью МО, построенного на глубоком обучении и сложных нейронных сетях. Сам же методологический подход (**Степень**) к автоматическому поиску уязвимостей описан с помощью следующих этапов: изучение процесса обнаружения уязвимостей, построение соответствующих моделей, их обучение и анализ результатов решения задачи поиска. МО авторы предлагают реализовывать на базе модели с подкреплением, и, в частности, с применением Q-обучения (**Реализация**), что позволит оптимизировать действия по поиску уязвимостей, аналогичные предпринимаемым человеком (**Задача**). Также приводится мнение экспертов касательно того, что ИИ, скорее всего, должен иметь рекомендательный характер, а окончательное решение касательно потенциальных уязвимостей должно приниматься экспертом (**Метод**).

Изобретение [17] содержит технический результат в виде повышения вероятности обнаружения уязвимостей в ПО (**Область**) за счет следующих действий (**Метод**): построения векторов атак на уязвимости; сбор датасета с «зараженным» кодом; построение AST и поиск в его элементах вектора атаки; составление путей между элементами; применение обученной на векторах атак модели МО для путей; итоговая классификация (**Задача**) уязвимостей с помощью алгоритма случайного леса, дерева решений или SVM (**Реализация**). Таким образом, хотя изобретение предполагает лишь результат творческой деятельности (**Степень**), однако логичность предложенных действий поиска уязвимостей позволяет говорить об их практической реализуемости.

Также в российском сегменте Интернета было найдено несколько свидетельств о регистрации программы для ЭВМ:

– в [18] содержится описание программы (**Степень**) для предсказания путем классификации (**Задача**) CVSS-метрик уязвимостей из базы данных (**Метод**) в формате CVE (**Область**); также оцениваются характеристики такого предсказания (точность, полнота и F-мера) на базе различных методов МО (**Реализация**) с помощью «исчерпывающего поиска кросс-валидации» [19];

Примечание: Кросс-валидация (перекрестная проверка) – это метод оценки моделей МО путем обучения нескольких из них на подмножествах доступных входных данных и их оценки на другом дополнительном подмножестве; используется для обнаружения неспособности распознать паттерн или шаблон.

– в [20] содержится решение по автоматическому поиску уязвимостей (**Степень**) в устройстве с сетевым доступом (**Область**), для чего о нем собирается информация, осуществляется поиск уязвимостей по БД (**Метод**), а также делается попытка их эксплуатации. Какие-либо детали касательно применения МО в реферате свидетельства отсутствуют (**Задача**, **Реализация**).

Поиск научных публикаций по указанному выше ключевому запросу показал критически малое количество отечественных исследований по сравнению с зарубежными; так, проанализированные в [7] релевантные работы российских ученых составляют бóльшую часть из всех, найденных в РИНЦ.

Систематизация результатов

Осуществим систематизацию 12 проведенных обзоров путем их характеризования с помощью следующего набора критериев:

К_1 – год публикации, позволяющий оценить тенденции в предметной области;

К_2 – область применения решения, заявленного авторами;

К_3 – основная идея метода поиска уязвимостей, описанная в абстрактных терминах (для сравнения с другими);

К_4 – стандартная задача, решаемая с помощью МО и лежащая в основе метода поиска: классификация, регрессия, выявление аномалий, кластеризация, обобщение, оптимизация действий;

К_5 – предлагаемая реализация МО: конкретный алгоритм, модель и т. п. (без учета ансамблевых метаалгоритмов); в случае их большого количества будем указывать наиболее эффективные (при наличии такой информации);

К_6 – степень реализации изложенных в публикации результатов: теоретическое предположение (теор.), доведение теории до состояния работающего прототипа (прот.) или полностью готовое практическое решение (практ.).

Результаты систематизации представлены в таблице 2 (использованы следующие сокращения: послед. – последовательность; множ. – множество; «---» – информация по критерию отсутствует; в скобках может указываться некоторая особенность соответствия критериям).

ТАБЛИЦА 2. Результаты систематизации обзоров научных публикаций

TABLE 2. Results of Systematizing Reviewing Scientific Publications

Название, ссылка	К_1	К_2	К_3	К_4	К_5	К_6
Automatic Detection and Correction of Vulnerabilities using Machine Learning [6]	2017	Web-сайты, БД	Сканирование и тестирование образца	регрессия, кластеризация	SVM	практ.
Current and Future Research of Machine Learning Based Vulnerability Detection [7]	2018	Программное обеспечение	Выделение признаков и поиск по ним	классификация, выявление аномалий, обобщение	Модель послед., BLSTM	теор.
Determining Web Application Vulnerabilities Using Machine Learning Methods [8]	2023	Web-сайты, БД	Выделение признаков и поиск по ним	классификация	Наивный Байес, SVM, дерево решений, случайный лес	практ.
A Machine Learning Based Approach to Identify SQL Injection Vulnerabilities [9]	2019	Программное обеспечение, БД	Выделение признаков и поиск по ним	классификация	MLP, RNN, BLSTM, CNN	прот.
An Empirical Study of High-Impact Factors for Machine Learning-Based Vulnerability Detection [11]	2020	Программное обеспечение	Выделение признаков и поиск по ним	классификация	GNN, RNN, BLSTM, CNN, преобразователь данных	прот.
Применение машинного обучения с подкреплением для автоматизированного поиска уязвимостей информационных систем [12]	2020	Web-сайты	Использование марковского процесса принятия решения	оптимизация действий	Модель МО с подкреплением (Q-обучение)	прот.
Обобщенная модель статического анализа программного кода на базе машинного обучения применительно к задаче поиска уязвимостей [13]	2020	Программное обеспечение	«Отражение» этапов поиска на задачи МО	Все	---	теор.
Поиск уязвимостей в машинном коде с помощью методов глубокого обучения [14]	2018	Программное обеспечение	Выделение признаков и поиск по ним	классификация	BLSTM	прот.

Название, ссылка	К_1	К_2	К_3	К_4	К_5	К_6
Использование ИИ в поиске уязвимостей в локальных сетях или веб-приложениях [16]	2023	Web-сайты, локальные сети	Рекомендательный характер (без детализации решения)	оптимизация действий	Модель МО с подкреплением (Q-обучение)	теор.
Способ и система выявления эксплуатируемых уязвимостей в программном коде [17]	2023	Программное обеспечение	Выделение признаков и поиск по ним	классификация	Случайный лес, дерево решений, SVM	теор.
Компонент анализа эффективности методов машинного обучения для предсказания значений метрик уязвимостей [18]	2023	Базы уязвимостей	Предсказание по БД	классификация	--- (множ.)	прот.
Прототип программного решения, реализующий перспективные технологии искусственного интеллекта применительно к тестированию на проникновение информационных систем [20]	2022	Сетевые приложения	Предсказание по БД	---	---	прот.

Исходя из результатов сравнительного анализа сделанных обзоров (см. таблицу 2), можно сделать следующие выводы по каждому из критериев. При этом критерии К_2 и К_4 будут оценены по балльной системе следующим образом – каждый 1 балл будет делиться между всеми значениями по критерию для одной публикации (например, значение «Web-сайты» по К_2 для [6] будет иметь 0.5 балла, а для [12] – 1 балл).

Во-первых, публикационная активность имеет характерный возрастающий тренд (несмотря на отдельные «провалы» в 2019 и 2022 годах), притом начавшийся в последние 5-10 лет; это можно видеть на гистограмме распределения количества проанализированных релевантных публикаций за соответствующий год (рисунок 1) с линейной аппроксимацией.

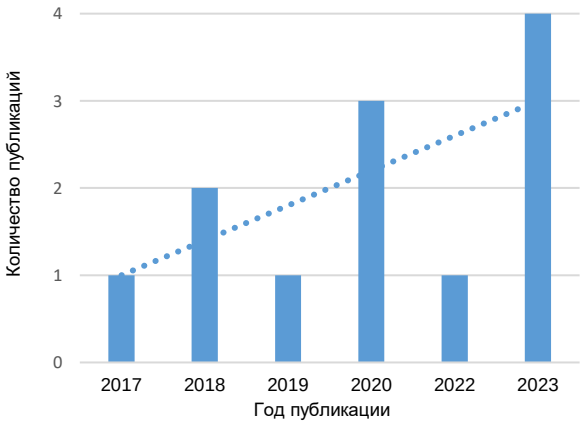

Рис. 1. Распределение количества публикаций по годам

Fig. 1. Distribution of the Number of Publications by Year

Можно предположить, что в будущем или тренд продолжит рост, поскольку необходимость выхода из «научно-технического тупика» по поиску уязвимостей станет еще актуальней, или наоборот – резкий спад, если будет найдено универсальное решение на базе МО (что представляется менее вероятным).

Во-вторых, областями, для которых авторы исследований предлагают применять свои решения, являются следующие (в порядке убывания «популярности», соответствующей баллам и указанной после рейтинга R):

- Программное обеспечение – $R = 5.5$;
- Web-сайты – $R = 2.5$;
- БД – $R = 1.5$;
- Сетевые приложения – $R = 1.0$;
- Локальные сети, Базы уязвимостей – $R = 0.5$

Таким образом, международная (как отечественная, так и зарубежная) научная общественность сходится во мнении, что применимость МО для поиска уязвимостей в ПО наиболее востребована.

В-третьих, сравнение идей (с учетом сложности такого процесса) позволяет поставить на первое место среди остальных выделение некоторых признаков в объекте исследования (например, поддеревьев AST для исходного кода) и осуществление по ним поиска уязвимостей, в основном, с помощью классификаторов МО – в 6 публикациях. Остальные же используемые идеи носят частный характер и не могут быть обобщены – например, применение марковских процессов [12], метрик из БД [18, 20], подход «серого» или «черного ящика» [6].

Т. е., каких-либо нестандартных и широко применяемых подходов для поиска уязвимостей обнаружено не было, поскольку в основном используется классическое сопоставление признаков с некоторым шаблоном (точным или вероятностным).

В-четвертых, в 1 публикации не указаны решаемые задачи, характерные для МО, а для 1 – указаны все. По остальным же работам распределение популярности, согласно баллам и указанной после рейтинга R, следующее:

– классификация – $R = 6\frac{1}{3}$;

– оптимизация действий – $R = 2$;

– регрессия, кластеризация – $R = 0.5$;

– обобщение, выявление аномалий – $R = \frac{1}{3}$.

Таким образом, наиболее часто применяемым способом в части МО для поиска уязвимостей является классификация. Впрочем, исходя из частоты решения задачи оптимизации действий (с помощью МО с подкреплением на базе Q-алгоритма), данный способ также может быть рассмотрен, как перспективный. Решение же других задач может оказаться востребованным для вспомогательных целей.

В-пятых, какого-либо существенного преимущества реализации МО не было выявлено, а их баллы распределились следующим образом: Модель МО с подкреплением (Q-обучение) – 2, LSTM (включая BLSTM) – 1.95, SVM – 1.5 балла, остальные – менее 0.5. Таким образом, можно говорить о выделении Q-алгоритмов и (B)LSTM-нейронной сети, как достаточно перспективных реализаций решения задачи поиска уязвимостей в ПО (хотя, следуя обзорам [12] и [16], Q-алгоритм не связан с этим напрямую). Как видно, все подходы к реализации МО имеют право на существование.

И, в-шестых, в 4-х работах описаны лишь теоретические выкладки, в 6-ти – предложен реально действующий прототип, и только в 2-х – исследование доведено до полноценного практического решения. Скорее всего, это означает, что или при переходе от прототипа к продукту возникают принципиальные сложности реализации, или же достигнутая эффективность поиска уязвимостей оказывается недостаточной для реального применения.

Концептуальная система поиска уязвимостей

На основании сделанного обзора публикаций, их систематизации, а также авторского опыта, предлагается (новая) концептуальная система поиска уязвимостей в программном обеспечении с применением МО (далее – Система), используемая экспертом по безопасности ПО (далее – Эксперт) и основанная на следующих принципах.

Принцип 1. В Системе должно применяться многостороннее (или многофункциональное) исследование образца программы: статический и динамический анализ, тестирование потенциально опасных мест, прогнозирование уязвимостей по метрикам и т. п.

Принцип 2. Должно использоваться сложное (многоэтапное, иерархическое, итеративное и т. п.) комбинирование известных методов сбора информации и поиска по ней уязвимостей.

Принцип 3. В основе всех частей-компонентов Системы (т. е. ее методов и управления ими) должно лежать МО.

Принцип 4. В случае невозможности гарантированного определения уязвимостей, их признаков, связанных с ними частей программы, Система может корректировать свою работу с помощью Эксперта (таким образом, Система относится к разряду поддерживающих принятие решений человеком).

Принцип 5. Информация, накапливаемая и используемая Системой, должна находиться в локальной базе данных, синхронизируемой с внешними хранилищами.

Принцип 6. Система должна давать рекомендации Эксперту касательно потенциальных уязвимостей.

Принцип 7. Все действия в Системе должны происходить на единой программно-аппаратной платформе в тесной связи друг с другом.

Таким образом, основная идея Системы заключается во всестороннем исследовании объекта (т. е. программы) различными базовыми методами, управляемыми настраиваемым метаалгоритмом. Вся собираемая информация при невозможности ее автоматической обработки должна корректироваться Экспертом, а результаты работы касательно найденных уязвимостей должны иметь рекомендательный характер, позволяя тем самым корректировать настройки Системы для повторного запуска. Также, метрики и другая информация об уязвимости должны браться не только из локальной БД (накапливая в результате работы), но и актуализироваться путем синхронизации с внешними БД (например, банком данных угроз ФСТЭК, NVD, CVE и др.).

Схема предлагаемой концептуальной Системы представлена на рисунке 2, где использованы следующие цветовые и иные обозначения: зеленый – объект исследования, синий – методы исследования, серый (с оттенками) – группировка или контейнер для элементов, оранжевый (с оттенками) – получаемые данные, красный – результат работы (т. е. уязвимости); желтый – использование МО; пометка «…» означает возможность расширения элементов группы.

Хотя схема является интуитивно понятной, тем не менее, дадим несколько принципиальных пояснений к ней путем указания соответствия принципов и частей схемы.

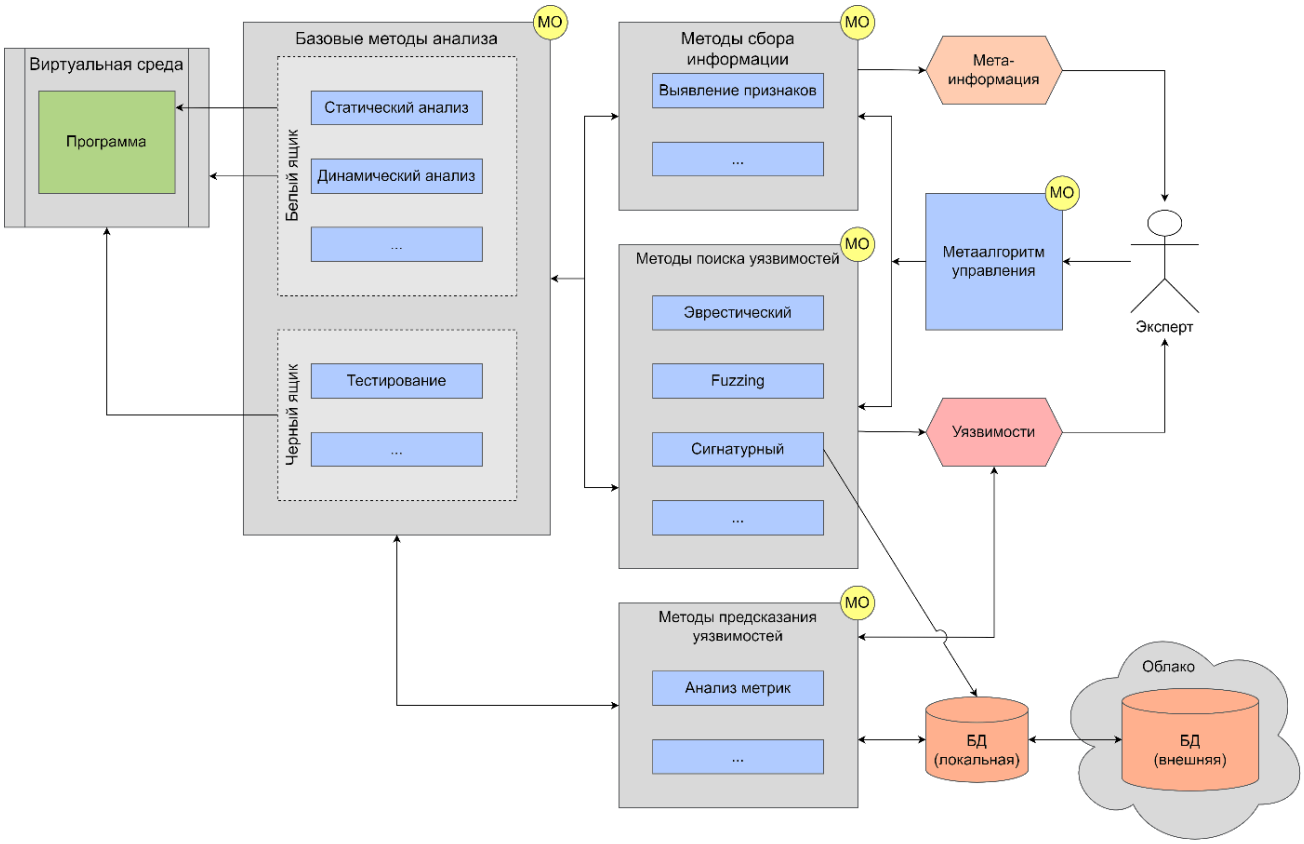

Рис. 2. Концептуальная система поиска уязвимостей в ПО с применением МО

Fig 2. A Conceptual System for Software Vulnerability Search Using Machine Learning

Принцип 1 учтен тем, что базовые методы анализа могут иметь различные походы («белый» и «черный ящик», а также их комбинацию), каждый из которых реализуем целым набором «инструментов» (статический и динамический анализ, тестирование выполнением и т. п.). При этом базовые методы могут взаимодействовать с объектом исследования напрямую (например, анализировать код программы) или через виртуальную среду (например, эмулируя выполнение программы).

Принцип 2 отражен с помощью комбинирования базовых методов при выполнении основных – сбора информации, поиска по ней уязвимостей, а также их предсказания (в том числе, по отдельным метрикам), которые могут оперировать целым набором базовых методов. Также, на схеме присутствует метаалгоритм, представляющий собой сложное решение по управлению другими методами, при этом в основе которого лежат не только эмпирические правила, но и обучаемые и тренируемые модели МО.

Выполнение Принципа 3 подтверждается тем, что практически все автоматические действия в Системе (кроме служебных, типа работы с БД) выполняются с применением МО (возможность чего была подтверждена в результатах рассмотренных исследований).

Соблюдение Принципа 4 достигается за счет прямого участия Эксперта в работе Системы путем предоставления ему собранной метаинформации о программе (в особенности, если такая не может дальше обрабатываться с необходимой точностью и полнотой), выявленных уязвимостях, их предсказаний на основании анализа обнаруженных или открытых метрик и т. п.

Для соответствия Принципу 5 в Системе предназначается локальная БД, синхронизуемая с глобальной (т. е. в «облаке»). Поскольку в основном в обеих БД хранится информация об уязвимостях (в том числе, их признаках в коде), то основное взаимодействие с БД осуществляют методы предсказания уязвимостей (например, по метрикам) и часть методов поиска уязвимостей (например, сигнатурный).

Поскольку согласно Принципу 6 Система предоставляет Эксперту лишь рекомендации (в смысле – вероятностные предположения) касательно результатов своей работы, то выходом автоматической части системы являются уязвимости, которые обрабатываются Экспертом и позволяют ему внести корректировки в алгоритмы работы Системы.

И наконец, все объекты, промежуточные данные, методы, БД (кроме глобальной) и сам Эксперт представляют собой взаимоувязанный человеко-управляемый программно-аппаратный комплекс по решению сложной научно-технической задачи, что подтверждает соответствие Принципу 7.

Заключение

В работе произведен обзор научных публикаций касательно возможностей применения МО для поиска уязвимостей в программах. Произведен сравнительный анализ обзоров (и их результатов), который позволил сделать систематизированные выводы касательно:

– актуализации исследования таких возможностей в последние годы;

– применимости к различным областям (хотя основной и является программное обеспечение);

– осуществления поиска уязвимостей (в основном, по их признакам в коде);

– решения задачи классификации вкупе с остальными, классическими для МО;

– преимуществ от применения той или иной реализации моделей и методов МО (установлено, что они отсутствуют);

– степени доведения идей исследователей до практических решений.

Как известно авторам, подобный обзор релевантных научных публикаций в РИНЦ (совместно с международными базами), содержащий предметно-ориентированное критериальное сравнение, выполнен впервые.

Учет достоинств, недостатков и особенностей изложенных в них решений, позволил авторам аргументированно декларировать 7 принципов построения концептуальной системы поиска уязвимостей в программном обеспечении с применение МО, а также собственно и синтезировать ее схему. Ее теоретическая значимость заключается в систематизации научного знания предметной области, а практическая состоит в возможности проектирования архитектуры такой системы и предпосылках к непосредственной реализации соответствующего программного решения.

Продолжением исследования должна стать оценка возможности реализации отдельных элементов Системы имеющимся инструментарием (а в случае его отсутствия – разработка нового).

Список источников

1. Романов Н.Е., Израилов К.Е., Покусов В.В. Система поддержки интеллектуального программирования: машинное обучение feat. быстрая разработка безопасных программ // Информатизация и связь. 2021. № 5. С. 7–17. DOI:10.34219/2078-8320-2021-12-5-7-16

2. Chavan A., Pimplikar S., Deshmukh A. An Overview of Machine Learning Techniques for Evaluation of Pavement Condition // Proceedings of the 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA, Goa, India, 08–09 October 2022). IEEE, 2022. PP. 139–143. DOI:10.1109/ICCCMLA56841.2022.9989164

3. Sathuluri M.R., Sahithi R., Sri P.N., Arshitha K. Machine Learning Approach to Design Fractal Antenna for 5G Applications // Proceedings of the 4th International Conference on Inventive Research in Computing Applications (ICIRCA, Coimbatore, India, 21–23 September 2022). IEEE, 2022. PP. 275–280. DOI:10.1109/ICIRCA54612.2022.9985480

4. Rana P., Gupta L. R., Dubey M.K., Kumar G. Review on evaluation techniques for better student learning outcomes using machine learning // Proceedings of the 2nd International Conference on Intelligent Engineering and Management (ICIEM, London, United Kingdom, 28–30 April 2021). IEEE, 2021. PP. 86–90. DOI:10.1109/ICIEM51511.2021.9445294

5. AlShehri Y., Ramaswamy L. SECOE: Alleviating Sensors Failure in Machine Learning-Coupled IoT Systems // Proceedings of the 21st International Conference on Machine Learning and Applications (ICMLA, Nassau, Bahamas, 2–14 December 2022). IEEE, 2022. PP. 743–747. DOI:10.1109/ICMLA55696.2022.00124

6. Tommy R., Sundeep G., Jose H. Automatic Detection and Correction of Vulnerabilities using Machine Learning // Proceedings of the International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC, Mysore, India, 08–09 September 2017). IEEE, 2017. PP. 1062–1065. DOI:10.1109/CTCEEC.2017.8454995

7. Jin Z., Yu Y. Current and Future Research of Machine Learning Based Vulnerability Detection // Proceedings of the Eighth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC, Harbin, China, 19–21 July 2018). IEEE, 2018. PP. 1562–1566. DOI:10.1109/IMCCC.2018.00322

8. Zhumabekova A., Matson E.T., Karyukin V., Zhumabekova K., Zhuandykov B., Ussatova O., et al. Determining Web Application Vulnerabilities Using Machine Learning Methods // Proceedings of the 19th International Asian School-Seminar on Optimization Problems of Complex Systems (OPCS, Novosibirsk, Moscow, Russian Federation, 14–22 August 2023). IEEE, 2023. PP. 136–139. DOI:10.1109/OPCS59592.2023.10275756

9. Zhang K. A Machine Learning Based Approach to Identify SQL Injection Vulnerabilities // Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (San Diego, USA, 11–15 November 2019). IEEE, 2019. PP. 1286–1288. DOI:10.1109/ASE.2019.00164

10. Le Q.V., Mikolov T. Distributed Representations of Sentences and Documents // Proceedings of the 31st International Conference on Machine Learning (PMLR, Beijing, China, 21–26 June 2014). 2014. Vol. 32. Iss. 2. PP. 1188–1196.

11. Zheng W., Gao J., Wu X., Xun Y., Liu G., Chen X. An Empirical Study of High-Impact Factors for Machine Learning-Based Vulnerability Detection // Proceedings of the IEEE 2nd International Workshop on Intelligent Bug Fixing (IBF, London, ON, Canada, 18–18 February 2020). IEEE, 2020. PP. 26–34. DOI:10.1109/IBF50092.2020.9034888

12. Выборнова О.Н., Рыжиков А.Н. Применение машинного обучения с подкреплением для автоматизированного поиска уязвимостей информационных систем // Математические методы в технике и технологиях – ММТТ. 2020. Т. 4. С. 110–113.

13. Буйневич М.В., Израилов К.Е. Обобщенная модель статического анализа программного кода на базе машинного обучения применительно к задаче поиска уязвимостей // Информатизация и связь. 2020. №. 2. С. 143–152. DOI:10.34219/2078-8320-2020-11-2-143-152

14. Демидов Р.А. Поиск уязвимостей в машинном коде с помощью методов глубокого обучения // IV межрегиональная научно-практическая конференция «Перспективные направления развития отечественных информационных технологий» (Севастополь, Российская Федерация, 18–22 сентября 2018). Севастопольский государственный университет, 2018. С. 237–238.

15. Bottou L. From machine learning to machine reasoning // Machine Learning. 2014. Vol. 94. Iss. 2. PP. 133–149. DOI:10.1007/s10994-013-5335-x

16. Осман С.Ш.О. Использование ИИ в поиске уязвимостей в локальных сетях или Веб-приложениях // IX международная научно-практическая конференция «Актуальные аспекты развития науки и общества в эпоху цифровой трансформации (шифр–МКАА)» (Москва, Российская Федерация, 25 июля 2023). Махачкала: Издательство АЛЕФ, 2023. С. 83–88.

17. Максимова А.А., Гончаренко Л.Х., Бачевский А.Е., Гуртова К.С., Умеренко Г.С., Анистратенко М.А. Способ и Система выявления эксплуатируемых уязвимостей в программном коде. Патент на изобретение RUS 2790005 C1 от 10.03.2022. Опубл. 14.02.2023.

18. Левшун Д.С. Компонент анализа эффективности методов машинного обучения для предсказания значений метрик уязвимостей. Свидетельство о регистрации программы для ЭВМ № RU 2023619249 от 05.05.2023. Опубл. 05.05.2023.

19. Celisse A. Optimal cross-validation in density estimation with the L^2-loss // The Annals of Statistics. 2014. Vol. 42. Iss. 5. PP. 1879–1910. DOI:10.1214/14-AOS1240

20. Кустаров Д.А., Сорокин Л.А., Трухачев А.А. Прототип программного решения, реализующий перспективные технологии искусственного интеллекта применительно к тестированию на проникновение информационных систем. Свидетельство о регистрации программы для ЭВМ № RU 2022682324 от 22.11.2022. Опубл. 29.11.2022.

References

1. Romanov N.E., Izrailov K.E., Pokussov V.V. Intelligent Programming Support System: Machine Learning Feat. Fast Development of Secure Programs. *Informatization and communication*. 2021;5:7–17. DOI:10.34219/2078-8320-2021-12-5-7-16

2. Chavan A., Pimplikar S., Deshmukh A. An Overview of Machine Learning Techniques for Evaluation of Pavement Condition. *Proceedings of the 4th International Conference on Cybernetics, Cognition and Machine Learning Applications, ICCCMLA, 08–09 October 2022, Goa, India)*. IEEE; 2022. p.139–143. DOI:10.1109/ICCCMLA56841.2022.9989164

3. Sathuluri M.R., Sahithi R., Sri P.N., Arshitha K. Machine Learning Approach to Design Fractal Antenna for 5G Applications. *Proceedings of the 4th International Conference on Inventive Research in Computing Applications, ICIRCA, 21–23 September 2022, Coimbatore, India*. IEEE; 2022. p.275–280. DOI:10.1109/ICIRCA54612.2022.9985480

4. Rana P., Gupta L. R., Dubey M.K., Kumar G. Review on evaluation techniques for better student learning outcomes using machine learning. *Proceedings of the 2nd International Conference on Intelligent Engineering and Management, ICIEM, 28–30 April 2021, London, United Kingdom*. IEEE; 2021. p.86–90. DOI:10.1109/ICIEM51511.2021.9445294

5. AlShehri Y., Ramaswamy L. SECOE: Alleviating Sensors Failure in Machine Learning-Coupled IoT Systems. *Proceedings of the 21st International Conference on Machine Learning and Applications, ICMLA, 2–14 December 2022, Nassau, Bahamas*. IEEE; 2022. p.743–747. DOI:10.1109/ICMLA55696.2022.00124

6. Tommy R., Sundeep G., Jose H. Automatic Detection and Correction of Vulnerabilities using Machine Learning. *Proceedings of the International Conference on Current Trends in Computer, Electrical, Electronics and Communication, CTCEEC, 08–09 September 2017, Mysore, India*. IEEE; 2017. p.1062–1065. DOI:10.1109/CTCEEC.2017.8454995

7. Jin Z., Yu Y. Current and Future Research of Machine Learning Based Vulnerability Detection. *Proceedings of the Eighth International Conference on Instrumentation & Measurement, Computer, Communication and Control, IMCCC, 19–21 July 2018, Harbin, China*. IEEE; 2018. p.1562–1566. DOI:10.1109/IMCCC.2018.00322

8. Zhumabekova A., Matson E.T., Karyukin V., Zhumabekova K., Zhuandykov B., Ussatova O., et al. Determining Web Application Vulnerabilities Using Machine Learning Methods. *Proceedings of the 19th International Asian School-Seminar on Optimization Problems of Complex Systems, OPCS, 14–22 August 2023, Novosibirsk, Moscow, Russian Federation*. IEEE; 2023. p.136–139. DOI:10.1109/OPCS59592.2023.10275756

9. Zhang K. A Machine Learning Based Approach to Identify SQL Injection Vulnerabilities. *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering, 11–15 November 2019, San Diego, USA*. IEEE; 2019. p.1286–1288. DOI:10.1109/ASE.2019.00164

10. Le Q.V., Mikolov T. Distributed Representations of Sentences and Documents. Proceedings of the 31st International Conference on Machine Learning, PMLR, 21–26 June 2014, Beijing, China. 2014;32(2):1188–1196.

11. Zheng W., Gao J., Wu X., Xun Y., Liu G., Chen X. An Empirical Study of High-Impact Factors for Machine Learning-Based Vulnerability Detection. *Proceedings of the IEEE 2nd International Workshop on Intelligent Bug Fixing, IBF, 18–18 February 2020, London, ON, Canada*. IEEE; 2020. p.26–34. DOI:10.1109/IBF50092.2020.9034888

12. Vybornova O.N., Ryzhikov A.N. Reinforcement Learning for Automated Vulnerability Search. *Matematicheskie metody v tekhnike i tekhnologiiakh – MMTT*. 2020;4:110–113.

13. Buinevich M.V., Izrailov K.E. Generalized model of software code`s static analysis based on machine learning for vulnerabilitys search. *Informatization and communication*. 2020;2:143–152. DOI:10.34219/2078-8320-2020-11-2-143-152

14. Demidov R.A. Vulnerability Search in Machine Code Using Deep Learning Approach. Proceedings of the IVth interregional Scientific and Practical Conference on Promising Directions for the Development of Domestic Information Technologies, 18–22 September 2018, Sevastopol, Russian Federation. Sevastopol State University Publ.; 2018. p.237–238.

15. Bottou L. From machine learning to machine reasoning. *Machine Learning*. 2014;94(2):133–149. DOI:10.1007/s10994-013-5335-x

16. Osman S.Sh.O. Using AI in Searching for Vulnerabilities in Local Networks or Web Applications. *IX International Scientific and Practical Conference on Current Aspects of the Development of Science and Society in the Era of Digital Transformation, Code – MCAA, 25 July 2023, Moscow, Russian Federation*. Makhachkala: ALEF Publ., 2023. p.83–88.

17. Maksimova A.A., Goncharenko L.Kh., Bachevsky A.E., Gurtova K.S., Umerenko G.S., Anistratenko M.A. *Method and System for Identifying Exploitable Vulnerabilities in Program Code*. Patent RF, no. 2790005 C1, 14.02.2023.

18. Levshun D.S. *Component for Analyzing the Effectiveness of Machine Learning Methods for Predicting the Values of Vulnerability Metrics*. Patent RF, no. 2023619249, 05.05.2023.

19. Celisse A. Optimal cross-validation in density estimation with the L^2-loss. *The Annals of Statistics*. 2014;42(5):1879–1910. DOI:10.1214/14-AOS1240

20. Kustarov D.A., Sorokin L.A., Trukhachev A.A. *A Prototype of a Software Solution That Implements Promising Artificial Intelligence Technologies in Relation to Penetration Testing Of Information Systems.* Patent RF, no. 2022682324, 29.11.2022.

Статья поступила в редакцию 01.12.2023; одобрена после рецензирования 14.12.2023; принята к публикации 18.12.2023.
The article was submitted 01.12.2023; approved after reviewing 14.12.2023; accepted for publication 18.12.2023.

Информация об авторе:

ЛЕОНОВ Николай Викторович
кандидат технических наук, доцент, начальник лаборатории Государственного научно-исследовательского института прикладных проблем
https://orcid.org/0009-0005-1295-5343

БУЙНЕВИЧ Михаил Викторович
доктор технических наук, профессор, профессор кафедры прикладной математики и информационных технологий Санкт-Петербургского университета государственной противопожарной службы МЧС России
https://orcid.org/0000-0001-8146-0022