УДК 004.056 *Научная статья* DOI: 10.18287/2223-9537-2024-14-3-408-420

Автоматическое оценивание эксплойтов на основе методов глубокого обучения

© 2024, Н.А. Бусько[1], Е.В. Федорченко[2], И.В. Котенко[2]✉

[1] *Санкт-Петербургский государственный электротехнический университет «ЛЭТИ», Санкт-Петербург, Россия*

[2] *Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (СПб ФИЦ РАН), Санкт-Петербург, Россия*

Аннотация

Оценивание и приоритизация программ, использующих уязвимости в программном обеспечении и применяемых для проведения кибератак на вычислительную систему (эксплойтов), является важным этапом эффективного реагирования на кибератаки. В данной работе предлагается методика автоматического оценивания эксплойтов, в которой на этапе проектирования выполняется обучение модели для классификации эксплойтов методами глубокого обучения, а на этапе эксплуатации обученная модель используется для вывода оценки критичности эксплойта. В основу методики положена гипотеза о том, что сложность применения эксплойта, последствия его применения и его оценка зависят от эксплуатируемой уязвимости и исходного кода эксплойта. Предложенная методика отличается от существующих применением для классификации эксплойтов модели *CodeBERT* на основе размеченного набора исходных кодов эксплойтов, а также разметкой исходных кодов эксплойтов в соответствии с оценками связанных уязвимостей по системе оценки уязвимостей *CVSS* (*Common Vulnerability Scoring System*) версии 2.0. Для экспериментальной оценки разработанной методики определены источники данных (база эксплойтов *Exploits-DB* и база уязвимостей *NVD*) и исходные данные для экспериментов, выполнен их статистический анализ, проведена экспериментальная оценка точности классификации эксплойтов. Полученные результаты могут использоваться при проектировании систем автоматического оценивания эксплойтов в рамках комплекса мер по мониторингу и повышению защищённости информационных систем.

Ключевые слова: *эксплойт, уязвимость, оценка, методика, модель, глубокое обучение, классификация данных, CodeBERT.*

Цитирование: *Бусько Н.А., Федорченко Е.В., Котенко И.В.* Автоматическое оценивание эксплойтов на основе методов глубокого обучения // Онтология проектирования. 2024. Т.14, №3(53). С.408-420. DOI: 10.18287/2223-9537-2024-14-3-408-420.

Финансирование: работа выполнена при поддержке гранта РНФ № 23-21-00498 в СПб ФИЦ РАН.

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Введение

В настоящее время можно выделить две тенденции, которые существенно влияют на развитие кибератак. Это повсеместная цифровизация, т.е. внедрение цифровых технологий, и развитие методов генеративного искусственного интеллекта (ИИ), т.е. ИИ, способного создавать новый контент. Повсеместная цифровизация приводит к увеличению поверхности кибератак (области их применения), а развитие методов генеративного ИИ – к автоматизации процесса проведения кибератак, в т.ч. этапа написания эксплойтов[1] для использования уяз-

[1] Экспло́йт (англ. *exploit*, эксплуатировать) — компьютерная программа, фрагмент программного кода или последовательность команд, использующие уязвимости в программном обеспечении и применяемые для проведения атаки на информационную систему.

вимостей. Чтобы эффективно противостоять киберугрозам, необходимо автоматизировать процессы выявления, оценивания и предотвращения успешного выполнения кибератак [1, 2].

В то время как вопросы автоматизации выявления кибератак, в т.ч. с использованием эксплойтов, рассматриваются во многих исследованиях, вопросы автоматизации оценивания эксплойтов остаются не исследованными. В данной работе выдвигается гипотеза, что сложность применения эксплойта, последствия его применения и его оценка зависят от эксплуатируемой уязвимости и исходного кода эксплойта. Исходя из этой гипотезы, предлагается подход к проектированию системы автоматического оценивания эксплойтов на основе оценок связанных с ними уязвимостей и последующему применению этой системы при реализации кибератак. Подход включает этапы сбора и подготовки набора данных, обработки данных, обучения моделей методами глубокого обучения (ГО).

Этап применения проектируемой системы включает вывод оценки для эксплойта с использованием обученной модели. В качестве набора данных выбран набор исходных кодов эксплойтов из базы эксплойтов *EDB*[2], размеченный в соответствии с оценками связанных уязвимостей по системе оценки уязвимостей *CVSS* (*Common Vulnerability Scoring System*) версии 2.0 [3]. Определено три способа разметки по классам в соответствии с оценками *CVSS*: по интегральной оценке критичности (выделено 3 класса), по промежуточным показателям *CVSS* (выделено 46 классов) и по атомарным показателям *CVSS* (выделено 67 классов). Для обработки данных использовались методы токенизации для перевода исходного кода эксплойтов в эмбеддинги[3]. Для обучения модели на основе трансформера [4] использовались методы ГО. Результатом классификации эксплойтов с применением обученной модели является оценка критичности эксплойта в соответствии с интегральной оценкой, промежуточными показателями и атомарными показателями *CVSS*. На практике такая система позволит оценивать новые эксплойты, не имеющие связи с уязвимостями, или те эксплойты, для которых связанные уязвимости не имеют оценки.

1 Методы и применение машинного и глубокого обучения

Решается задача классификации эксплойтов, целью которой является выбор метки для каждого экземпляра в наборе данных на основе входных признаков. Рассмотрены известные методы машинного обучения (МО) и ГО от классических алгоритмов МО до глубоких нейронных сетей (НС). К классическим алгоритмам относятся логистическая регрессия [5], метод опорных векторов (*Support Vector Machine, SVM*) [6], деревья решений и ансамбли [7], стохастический градиентный бустинг[4] [8], случайный лес (*Random Forest, RF*) [9], наивный Байесовский классификатор [10]. К методам ГО относятся искусственные НС (*Artificial Neural Networks, ANN*) [11], свёрточные НС (*Convolutional Neural Networks, CNN*) [12], рекуррентные НС (*Recurrent neural network, RNN*) [13, 14], долгая краткосрочная память (*Long short-term memory, LSTM*) [15], модели на основе трансформеров [4] и др.

Методы МО и ГО применяются для решения различных задач, связанных с анализом исходного кода. Статья [16] посвящена задаче автоматического определения языка программирования исходного кода с использованием методов МО. В ней для классификации используется метод максимальной энтропии. Объединённая грамматика создаётся автоматически на основе обучающего набора данных, затем эта грамматика используется для извлечения наиболее представительных грамматических конструкций, которые служат признаками для

[2] *Exploit Database*. https://www.exploit-db.com/.
[3] Эмбеддинг (англ. *embedding,* встраивание) — вектор, представленный в виде массива чисел, который получается в результате преобразования данных (например, текста).
[4] Бустинг (англ. *boosting,* усиление) — ансамблевый метаалгоритм МО.

классификации. Достигается точность классификации 99 % при определении 29 языков программирования. Анализ исходного кода применяется также в задачах рефакторинга кода. В работе [17] глубокие НС используются для атрибуции исходного кода.

Анализ исходного кода активно применяется в области информационной безопасности. В частности, для обнаружения и классификации вредоносного программного обеспечения [18, 19] и для обнаружения уязвимостей [20–23].

Наиболее близким к данной работе является исследование [24], в котором используются методы ИИ и обработки естественного языка для автоматической категоризации уязвимых фрагментов исходного кода в соответствии с общей системой перечисления слабых мест (*Common Weakness Enumeration, CWE*[5]). В этой работе используются методы «мешок слов» (*Bag-of-Words*) и последовательности токенов для представления исходного кода, *word2vec, fastText, BERT* и *CodeBERT* для векторизации слов, *RF, SVM* и трансформеры для обучения моделей. Результаты исследования показали эффективность применения контекстных эмбеддингов предобученных моделей на базе трансформеров для классификации уязвимостей в исходном коде.

Ряд работ посвящён вопросам оценивания уязвимостей. В работе [25] показано применение методов обработки естественного языка для автоматического оценивания уязвимостей на основе их описаний, в работе [26] *CVSS* векторы для уязвимостей прогнозируются путём применения МО к их текстовым описаниям.

2 Система автоматического оценивания эксплойтов с использованием методов ГО

Для проектирования системы автоматического оценивания эксплойтов разработаны методики: сбора и подготовки набора данных; обработки набора данных; обучения моделей классификации эксплойтов для их оценивания. Этап применения системы включает вывод оценки для эксплойта с использованием обученной модели (рисунок 1).

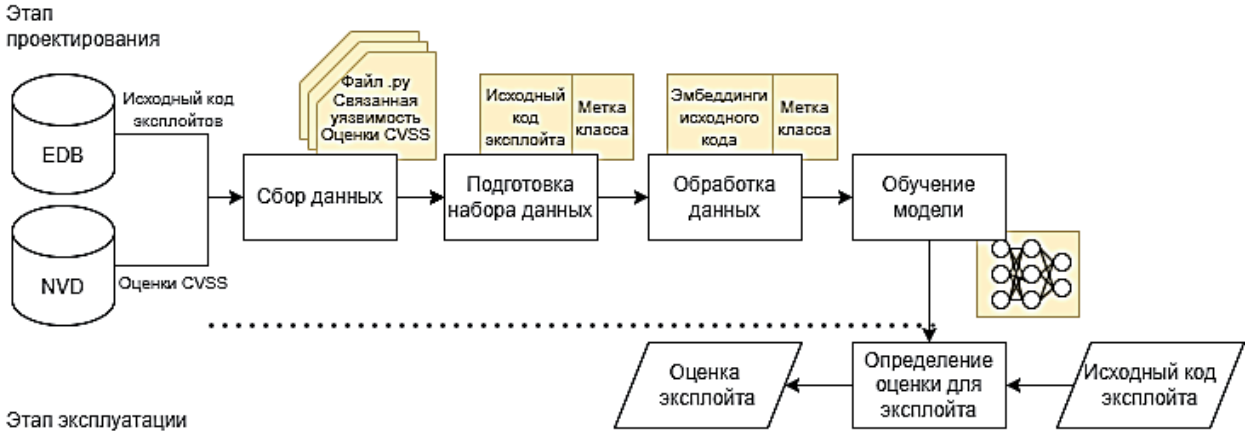

Рисунок 1 – Архитектура системы автоматического оценивания эксплойтов

2.1 Сбор и подготовка набора данных

В качестве набора данных выбран набор исходных кодов эксплойтов из *EDB*, размеченный в соответствии с оценками связанных уязвимостей по *CVSS* [3]. Эксплойт в *EDB* описывается следующими полями: уникальный идентификатор; ссылка на уязвимость в формате

[5] *Common Weakness Enumeration*. https://cwe.mitre.org/index.html.

открытых уязвимостей и дефектов *CVE*[6]; автор эксплойта; тип атаки; платформа, для которой предназначен эксплойт; дата публикации эксплойта; исходный код эксплойта.

Всего *EDB* содержит 52023 эксплойта. Собрано и рассмотрено 48316 эксплойтов. Для этого набора данных выполнен анализ по значениям следующих полей: используемый язык программирования; целевая платформа; год создания эксплойта.

Оценка распределения эксплойтов по используемому языку программирования показала, что наиболее используемыми языками являются *Python, C, Perl, Ruby* и *HTML*. Для исследования выбраны 4139 эксплойтов, написанные на языке программирования *Python*.

Оценка распределения эксплойтов по целевой платформе показала, что наибольшее количество эксплойтов направлено на удалённые серверы и веб-приложения. Наибольшее количество эксплойтов было создано в 2010 г.

Для разметки исходных кодов эксплойтов выбраны оценки связанных уязвимостей по системе оценки уязвимостей *CVSS* версии 2.0, представленные в базе данных уязвимостей *NVD*[7]. Система оценки уязвимостей *CVSS* версии 2.0 включает базовые, временные и контекстные показатели, а в *NVD* представлены только базовые показатели.

CVSS версии 2.0 включает следующие базовые показатели, для которых в скобках указаны их возможные значения [27]: «Вектор доступа» («Локальный», «Смежная сеть», «Сетевой»), «Сложность доступа» («Высокий», «Средний», «Низкий»), «Аутентификация» («Множество», «Один», «Нет»), «Влияние на конфиденциальность» («Нет», «Частичное», «Полное»), «Влияние на целостность» («Нет», «Частичное», «Полное»), «Влияние на доступность» («Нет», «Частичное», «Полное»).

На основе показателей «Вектор доступа», «Сложность доступа» и «Аутентификация» рассчитывается интегральный показатель «Эксплуатируемость» («Недоказанная», «Концептуальная», «Функциональная», «Высокая»). На основе показателей «Влияние на конфиденциальность», «Влияние на целостность» и «Влияние на доступность» рассчитывается интегральный показатель «Ущерб». На основе показателей «Эксплуатируемость» и «Ущерб» рассчитывается базовая оценка уязвимости по *CVSS*, или «Критичность» уязвимости («Высокая», «Средняя», «Низкая»).

Общее количество эксплойтов, написанных на языке *Python* и имеющих ссылки на уязвимости, - 1647. Размер обучающей выборки, полученный суммированием количества символов в файлах с исходным кодом эксплойтов, составляет 26326807 символов. Из 1647 эксплойтов оценки по стандарту *CVSS* версии 2.0 обнаружены для 1565 записей.

Определено три способа разметки эксплойтов по классам в соответствии со значениями метрик *CVSS* связанных уязвимостей: по базовой оценке уязвимости «Критичность», по промежуточным показателям *CVSS* «Эксплуатируемость» и «Ущерб», по атомарным показателям *CVSS* «Аутентификация», «Влияние на конфиденциальность», «Влияние на целостность», «Влияние на доступность», «Вектор доступа» и «Сложность доступа». Каждый выделенный класс эксплойтов соответствует уникальной комбинации значений метрик и определяет степень ущерба, который может быть причинён использованием эксплойта.

2.2 Обработка данных

Для обработки данных использовались методы токенизации для перевода исходного кода эксплойтов в эмбеддинги. В данном исследовании для предобработки данных использовался токенизатор *RobertaTokenizer* от модели *CodeBERT* [28]. Результатом этапа обработки данных являются эмбеддинги исходных кодов и метки соответствующих классов.

[6] *Common Vulnerabilities and Exposures*. https://www.cve.org/

[7] *National Vulnerability Database*. https://nvd.nist.gov/.

2.3 Обучение моделей

Для обучения модели использовались методы ГО, в частности модель *CodeBERT*. *CodeBERT* является расширением модели-трансформера *BERT* (*Bidirectional Encoder Representations from Transformers*) [29]. Она представляет собой предварительно обученную модель для языков программирования (*programming language, PL*) и естественных языков (*natural language, NL*). *CodeBERT* была обучена на парах *NL-PL* на шести языках программирования (*Python, Java, JavaScript, PHP, Ruby, Go*). *CodeBERT* включает 12 слоёв и 125 млн параметров [28].

Схема работы *CodeBERT* представлена на рисунке 2:

1) на вход подаётся пара «исходный код» - «комментарий»;
2) случайным образом выбирается часть токенов, которые заменяются специальным токеном [*MASK*];
3) модель *CodeBERT* обучается прогнозировать замаскированные токены;
4) к необработанным предсказаниям модели применяется функция активации *Softmax*, которая преобразует их в вектор вероятностей.

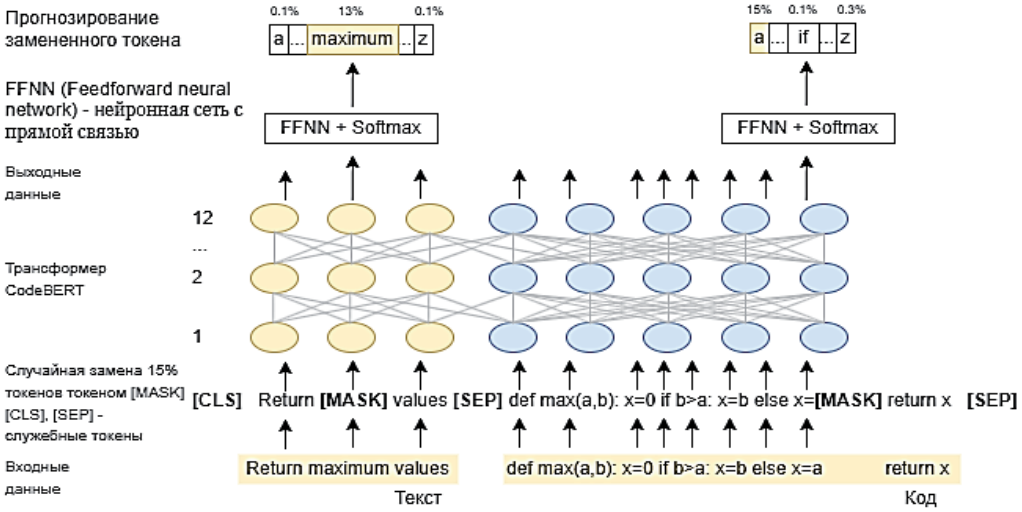

Рисунок 2 – Схема работы *CodeBERT*

Процесс обучения модели осуществляется путём оптимизации параметров с целью минимизации функции потерь и включает следующие этапы:

1) *Прямое распространение сигнала.* Токенизированные данные подаются на вход модели, и вычисляются выходные представления для каждой последовательности. При этом вначале входные идентификаторы токенов преобразуются в эмбеддинги – многоразмерные матрицы исходных токенов, к которым применена маска внимания, указывающая, какие токены в последовательности являются значимыми, а какие – нет. Эмбеддинги проходят через несколько слоёв энкодера, внутри которых выполняются операции: самовнимания, нормализация, позиционные и линейные преобразования. В результате этих операций на выходе энкодера формируются логиты – необработанные предсказания модели. Далее применяется функция активации *Softmax*, которая преобразует вектор логитов в вектор вероятностей.
2) *Вычисление функции потерь.* Разница между выходными представлениями модели и истинными метками оценивается с помощью функции потерь. В исследовании применяется функция потерь кросс-энтропии *LogSoftmax*, которая вычисляет логарифмы вероятностей, а затем суммирует отрицательные логарифмы. Далее применяется отрицательная логарифмическая вероятность для вычисления отрицательного логарифма вероятности правильного класса. Затем вычисляется кросс-энтропия между предсказанными значениями и истинными метками.
3) *Обратное распространение ошибки.* Градиенты функции потерь вычисляются относительно параметров модели. Они впоследствии используются для обновления весов модели.
4) *Обновление параметров.* Параметры модели обновляются с использованием метода оптимизации. За обновление параметров модели (весов НС) за счёт минимизации функции потерь на каждом шаге обучения

отвечает оптимизатор, т.е. с его помощью определяется, каким образом и на сколько должны изменяться веса НС на каждом шаге обучения для улучшения предсказательной способности модели. В данном исследовании используется оптимизатор *AdamW* (*Adaptive Moment Estimation with Weight Decay*) [30]. Кроме того обновляется скорость обучения, т.е. насколько сильно изменяются веса на каждом шаге обучения, с использованием планировщика. Проводится оценка модели по текущим параметрам: если текущая точность превышает ранее записанную, то она обновляется.

Обученная НС проходит этапы валидации и тестирования. Результатом классификации эксплойтов с использованием обученной модели является оценка критичности эксплойта в соответствии с интегральной оценкой, промежуточными показателями и атомарными показателями *CVSS*.

3 Экспериментальная оценка предложенного подхода

Для проведения экспериментов реализован прототип на языке *Python*[8]. Архитектура прототипа представлена на рисунке 3.

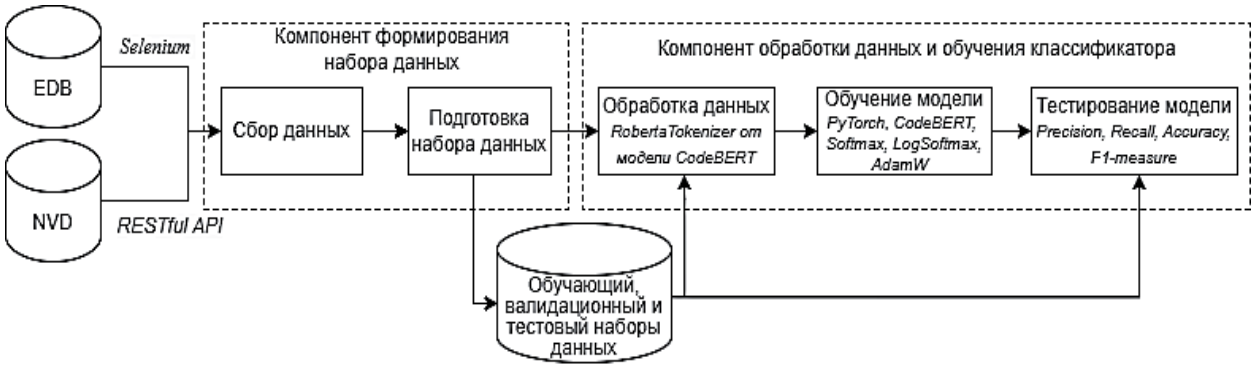

Рисунок 3 – Архитектура прототипа системы оценивания эксплойтов

3.1 Формирование набора данных

Входными данными были исходные коды эксплойтов, написанные на языке *Python*, из базы *EDB*, и оценки *CVSS* версии 2.0 для связанных с эксплойтами уязвимостей из базы *NVD*. Для сбора исходных кодов эксплойтов из *EDB* использовался фреймворк *Selenium* для *Python*. Для сбора данных об оценках *CVSS* версии 2.0 из *NVD* задействовался предоставляемый *NVD RESTful API*[9]. Результатом данного этапа является набор исходных кодов эксплойтов и соответствующие им *CVSS* оценки версии 2.0 (всего 1565 записей). На следующем этапе производилась разметка набора данных с учётом классов эксплойтов, выделенных в соответствии со значениями метрик *CVSS* связанных уязвимостей:

(1) по интегральной оценке критичности связанной уязвимости «Критичность» (выделено 3 класса);
(2) по промежуточным показателям *CVSS* «Эксплуатируемость» и «Ущерб» (выделено 46 классов);
(3) по атомарным показателям *CVSS* «Аутентификация», «Влияние на конфиденциальность», «Влияние на целостность», «Влияние на доступность», «Вектор доступа» и «Сложность доступа» (67 классов).

Для повышения надёжности анализа введены пороговые значения количества экземпляров в классе: 35, 100 и 200. При пороговом значении 35 получено 9 уникальных классов для первого варианта классификации, 8 для второго и 3 для третьего. Для порога 100 классов получено по 4 класса для первых двух вариантов и 2 класса для третьего. При пороге в 200 экземпляров выявлено 2, 3 и 2 уникальных класса, соответственно. Каждый выделенный класс эксплойтов соответствует уникальной комбинации значений метрик и определяет степень ущерба, который может быть причинён использованием эксплойта.

[8] https://github.com/nbusko/exploits_classifier_pipeline.
[9] https://nvd.nist.gov/developers/data-sources.

С целью валидации модели собранный набор данных разделён на обучающую, тестовую и валидационную выборки в соотношении 80%, 15% и 5%. В результате сформировано 9 обучающих и 9 тестовых наборов данных. Фрагмент набора данных для классов, выделенных по интегральной оценке критичности связанной уязвимости «Критичность», представлен в таблице 1: в столбце «№» отображается порядковый номер записи, в столбце «Code» – фрагмент исходного кода эксплойта, в столбце «v2severity» – значение интегральной оценки «Критичность», в столбце «target» – метка класса.

Таблица 1 – Фрагмент набора данных для классов, выделенных по интегральной оценке критичности связанной уязвимости «Критичность»

№	Code	v2severity	target
0	#!/usr/bin/python\n# win7-crash.py:\n# Trigger a remote kernel crash on Win7 and server 2008R2 (infinite loop)\n# Crash in KeAccumulateTicks() due to NT_ASSERT()/DbgRaiseAssertionFailure() caused by an \n#infinite loop.\n#NO BSOD, YOU GOTTA PULL THE PLUG.\n#To trigger it fast: from the target: \\this_script_ip_addr\BLAH. instantly crash\n#Author: Laurent Gaffii¿½\n\nimport SocketServer\n\npac...	HIGH	0
1	#!/usr/bin/python\n# HP Power Manager Administration Universal Buffer Overflow Exploit\n# CVE 2009-2685\n# Tested on Win2k3 Ent SP2 English, Win XP Sp2 English\n# Matteo Memelli ryujin _A-T__ offensive-security.com\n# www.offensive-security.com\n# Spaghetti & Pwnsauce - 07/11/2009\n#\n# ryujin@bt:~$ ./hppowermanager.py 172.16.30.203\n# HP Power Manager Administration Universal Buffer Overflow…	HIGH	0
2	+#!/usr/bin/python\n#\n# ## \n#\n# Exploit Title : Serenity Audio Player Playlist (m3u) BOF\n# Discovered by\t : Rick from Corelan Team (ricks2600[at]gmail[dot]com)\n# Author : mr_me\n# Author contact : seeleymagic[at]hotmail[dot]com\n# Date : nov 24th, 2009\n# Type : local and rem…	HIGH	0
…	…	…	…
1562	#!/usr/bin/python3\n\n# Exploit Title: Tenda N300 F3 12.01.01.48 - Malformed HTTP Request Header Processing \n# Shodan Dork: http favicon.hash:-2145085239 http.title:"Tenda | LOGIN"\n# Date: 09/03/2023\n# Exploit Author: @h454nsec\n# Github: https://github.com/H454NSec/CVE-2020-35391\n# Vendor Homepage: https://www.tendacn.com/default.html\n# Product Link: https://www.tendacn.com/product/f3.ht...	Low	2
1563	# Exploit Title: NCH Express Invoice - Clear Text Password Storage and Account Takeover\n# Google Dork:: intitle:Expressinvoice - Login\n# Date: 07/Apr/2020\n# Exploit Author: Tejas Nitin Pingulkar (https//cvewalkthrough.com/)\n# Vendor Homepage: https://www.nchsoftware.com/\n# Software Link: http://www.oldversiondownload.com/oldversions/express-8-05-2020-06-08.exe\n# Version: NCH Express Inv...	Low	2
1564	# WinFTP v2.3.0 DoS exploit\n# WinFTP URL - http://mww.wftpserver.com/\n# DoS'ed when try to send data\n# (x)dmnt\n# -*- coding: windows-1252 -*-\n\nimport socket\nimport time\nimport sys\n\nPORT = 21\n\ndef help_info():\n print ("Usage: winftp <host> <login> <password>\n")\n print ("Note: anonymous is enought\n")\n\ndef conn(hostname, username, passwd):\n sock = socket socket(socket…	Low	2

3.2 Обработка данных и обучение классификатора

По результатам предыдущего этапа сформирован словарь с данными, содержащий исходный код эксплойта и метку класса. При обработке данных для перевода исходного кода эксплойтов в эмбеддинги выполняются следующие этапы:

1) извлечение исходного кода из элемента словаря;
2) токенизация кода с использованием *RobertaTokenizer*, после разделения строки на отдельные токены выполняется усечение списка токенов до заданного размера размера *block_size* (в данном исследовании – 2);
3) добавление специальных токенов – в начало каждой входной текстовой последовательности помещаются токены [*CLS*], для разделения разных частей текста в одной последовательности и для обозначения конца последовательности используются токены [*SEP*];
4) преобразование токенов в идентификаторы, т.е. в соответствующие им числовые идентификаторы из словаря токенизатора;
5) добавление отступа – список идентификаторов дополняется до размера блока.

Выходными данными этапа обработки данных является список токенов, список идентификаторов и метка класса. Обработанные данные используются для обучения классификатора. В исследовании применялась библиотека *PyTorch* для обучения модели *CodeBERT*. Основной цикл обучения классификатора выполняется следующим образом:

1) градиенты всех параметров модели инициализируются нулями;

2) инициализируется цикл, который будет выполняться в течение заданного числа эпох (в данном исследовании этот параметр равен 7) и включает этапы обучения классификатора, описанные в разделе 2:
   2.1) прямое распространение сигнала (получение вектора вероятностей для входной последовательности);
   2.2) расчёт градиентов функции потерь *LogSoftmax* относительно параметров модели, значения этих градиентов впоследствии используются для обновления весов модели;
   2.3) обновление параметров модели согласно градиентам с использованием оптимизатора *AdamW*;
   2.4) обнуление градиентов оптимизатора;
   2.5) обновление скорости обучения с использованием планировщика;
3) проводится оценка модели по текущим параметрам на валидационном наборе данных: если текущая точность превышает ранее записанную, то она обновляется до текущей.

3.3 Тестирование и оценка эффективности работы классификатора

В ходе исследования обучены 9 моделей. При обучении использовались наборы данных, отличающиеся выделенными в соответствии с показателями *CVSS* версии 2.0 классами и пороговыми значениями количества примеров для каждого класса.

В таблице 2 приведены названия моделей и соответствующие показатели *CVSS* версии 2.0 и пороги. Названия моделей сформированы следующим образом: «*models*_{пороговое значение примеров}_{вариант компоновки метрик *CVSS*}_{количество уникальных классов}».

Таблица 2 – Характеристики наборов данных, использовавшихся при обучении моделей

Название модели	Показатели *CVSS*	Порог	Количество уникальных классов
models_200_var_1_unique2	«Аутентификация», «Вектор доступа», «Сложность доступа», «Влияние на конфиденциальность», «Влияние на целостность», «Влияние на доступность»	200	2
models_200_var_2_unique3	«Эксплуатируемость», «Ущерб»	200	3
models_200_var_3_unique2	«Критичность»	200	2
models_35_var_1_unique9	«Аутентификация», «Вектор доступа», «Сложность доступа», «Влияние на конфиденциальность», «Влияние на целостность», «Влияние на доступность»	35	9
models_35_var_2_unique8	«Эксплуатируемость», «Ущерб»	35	8
models_35_var_3_unique3	«Критичность»	35	3
models_100_var_1_unique4	«Аутентификация», «Вектор доступа», «Сложность доступа», «Влияние на конфиденциальность», «Влияние на целостность», «Влияние на доступность»	100	4
models_100_var_2_unique4	«Эксплуатируемость», «Ущерб»	100	4
models_100_var_3_unique2	«Критичность»	100	2

Обученные классификаторы протестированы на данных, которые не были использованы в обучении моделей. Результаты приведены в таблице 3. Из таблицы 3 видно, что модель *models_100_var_2_unique4* имеет лучшие значения по всем показателям.

Предложенная методика автоматического оценивания эксплойтов методами ГО на основе размеченного в соответствии с оценками связанных уязвимостей по системе *CVSS* версии 2.0 набора исходных кодов эксплойтов положена в основу проектирования системы автоматического оценивания эксплойтов. С её помощью подтверждена гипотеза о том, что сложность применения эксплойта и последствия его применения, а значит и его оценка, зависят от эксплуатируемой уязвимости и исходного кода эксплойта.

Таблица 3 – Результаты тестирования обученных моделей

№	Название модели	Точность (Precision)	Полнота (Recall)	*F1*-мера (F1-measure)	Точность (Accuracy)
0	*models_200_var_1_unique2*	0.648148	0.648148	0.648148	0.649123
1	*models_200_var_2_unique3*	0.589514	0.587402	0.586655	0.582278
2	*models_200_var_3_unique2*	0.693491	0.676142	0.674583	0.686275
3	*models_35_var_1_unique9*	0.462370	0.481367	0.462824	0.519380
4	*models_35_var_2_unique8*	0.567976	0.454073	0.479167	0.536765
5	*models_35_var_3_unique3*	0.484799	0.481756	0.480915	0.732484
6	*models_100_var_1_unique4*	0.576822	0.539749	0.552085	0.565657
7	*models_100_var_2_unique4*	**0.768646**	**0.780400**	**0.765575**	**0.750000**
8	*models_100_var_3_unique2*	0.710924	0.710924	0.710924	0.712418

Заключение

В работе предложен подход к проектированию системы автоматического оценивания эксплойтов, в основе которого лежит обучение модели для классификации исходных кодов эксплойтов по оценкам связанных уязвимостей методами ГО. Проанализированы базы данных эксплойтов и уязвимостей и собран набор исходных кодов эксплойтов для классификации. Предложена оригинальная разметка набора данных в соответствии с показателями *CVSS* применяемых в эксплойтах уязвимостей. Обучено 9 моделей на базе *CodeBERT* с использованием трёх вариантов выделения классов при разметке набора данных и разным пороговым значением примеров в классе. По результатам проведённых экспериментов лучшие значения по всем показателям получены на модели с пороговым значением примеров в классе – 100, вариантом компоновки метрик *CVSS* – «Эксплуатируемость», «Ущерб» и количеством уникальных классов – 4. В результате исследования подтверждена гипотеза, что сложность применения эксплойта, последствия его применения и оценка зависят от эксплуатируемой уязвимости и исходного кода эксплойта.

Полученные результаты могут использоваться для классификации и оценки эксплойтов с открытым исходным кодом и новых эксплойтов, обнаруженных системами обнаружения вторжений, что позволит приоритизировать обнаруженные атаки при принятии решений о необходимости реагирования.

Список источников

[1] ***Котенко И.В., Саенко И.Б., Полубелова О.В., Чечулин А.А.*** Технологии управления информацией и событиями безопасности для защиты компьютерных сетей // Проблемы информационной безопасности. Компьютерные системы. 2012. № 2. С. 57-68.

[2] ***Котенко И.В., Ушаков И.А.*** Технологии больших данных для мониторинга компьютерной безопасности // Защита информации. Инсайд. 2017. № 3 (75). С. 23-33.

[3] ***Mell P., Scarfone K., Romanosky S.*** A complete guide to the Common Vulnerability Scoring System. Version 2.0. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51198.

[4] ***Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Polosukhin I., Kaiser Ł.*** Attention is all you need // In: Proc. of the 31st Int. Conf. on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 2017. P.6000–6010.

[5] ***Akram Z., Majid M., Habib S.*** A systematic literature review: usage of logistic regression for malware detection // In: Proc. of the 2021 Int. Conf. on Innovative Computing (Lahore, Pakistan, 2021). 2021. P.1-8. DOI: 10.1109/ICIC53490.2021.9693035.

[6] ***Ghanem K., Aparicio-Navarro F. J., Kyriakopoulos K. G., Lambotharan S., Chambers J. A.*** Support Vector Machine for network intrusion and cyber-attack detection // In: Proc. of the 2017 Sensor Signal Processing for Defence Conf. (London, UK, 2017). 2017. P.1-5. DOI: 10.1109/SSPD.2017.8233268.

[7] ***Breiman, L***. Bagging predictors // Mach. Learn. 1996. Vol. 24. P.123–140.

[8] ***Vu Q. H., Ruta D., Cen L.*** Gradient boosting decision trees for cyber security threats detection based on network events logs // In: Proc. of the 2019 IEEE Int. Conf. on Big Data (Los Angeles, CA, USA, 2019). 2019. P.5921-5928. DOI: 10.1109/BigData47090.2019.9006061.

[9] ***Choubisa M., Doshi R., Khatri N., Kant Hiran K.*** A simple and robust approach of random forest for intrusion detection system in cyber security // In: Proc. of the 2022 Int. Conf. on IoT and Blockchain Technology (Ranchi, India, 2022). 2022. P.1-5. DOI: 10.1109/ICIBT52874.2022.9807766.

[10] ***Mukherjee S., Sharma N.*** Intrusion detection using naive Bayes classifier with feature reduction // Procedia Technology. 2012. Vol. 4. P.119–128. https://doi.org/10.1016/j.protcy.2012.05.017. 455.

[11] ***Prajoy P., Subrato B., Rubaiyat M. M., Pinto P., Utku K.*** Artificial Neural Network for Cybersecurity: A Comprehensive Review // Journal of Information Assurance & Security. 2021. Vol. 16. Issue 1. P.10.

[12] ***Krizhevsky A., Sutskever I., Hinton G.E.*** ImageNet classification with deep convolutional neural networks // Advances in Neural Information Processing Systems. 2012. P.15–30. https://arxiv.org/abs/1207.0580.

[13] ***Bengio Y., Ducharme R., Vincent P., Jauvin, C.*** A Neural Probabilistic Language Model // Journal of Machine Learning Research. 1994. P.1–18. https://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf.

[14] ***Pascanu R., Mikolov T., Bengio, Y.*** On the difficulty of training recurrent neural networks // In: Proc. of the 30th Int. Conf. on Machine Learning. 2013. P.1–12. https://arxiv.org/abs/1211.5063.

[15] ***Laghrissi F., Douzi S., Douzi K., Hssina B.*** Intrusion detection systems using long short-term memory (LSTM) // J Big Data. 2021. Vol. 8. Issue 65. https://doi.org/10.1186/s40537-021-00448-4.

[16] ***Shaul Z., Holzem C.*** Machine learning based source code classification using syntax oriented features. *ArXiv* abs/1703.07638. 2017.

[17] ***Mateless R., Tsur O., Moskovitch R.*** Pkg2Vec: Hierarchical package embedding for code authorship attribution // Future Gener. Comput. Syst. 2021. Vol. 116. P.49–60. https://doi.org/10.1016/j.future.2020.10.020.

[18] ***Sebastio S., Baranov E., Biondi F., Decourbe O., Given-Wilson T., Legay A., Puodzius C., Quilbeuf J.*** Optimizing symbolic execution for malware behavior classification // Comput. Secur. 2020. Vol. 93. P.101775. https://doi.org/10.1016/j.cose.2020.101775.

[19] ***Naeem H., Ullah F., Naeem M. R., Khalid S., Vasan D., Jabbar S., Saeed S.*** Malware detection in industrial internet of things based on hybrid image visualization and deep learning model // Ad Hoc Netw. 2020. Vol. 105. P.102154. https://doi.org/10.1016/j.adhoc.2020.102154.

[20] ***Sun H., Cui L., Li L., Ding Z., Hao Z., Cui J., Liu P.*** VDSimilar: Vulnerability detection based on code similarity of vulnerabilities and patches // Comput. Secur. 2021. Vol. 110. P.102417. https://doi.org/10.1016/j.cose.2021.102417.

[21] ***Moti Z., Hashemi S., Karimipour H., Dehghantanha A., Jahromi A. N., Abdi L., Alavi F.*** Generative adversarial network to detect unseen Internet of Things malware // Ad Hoc Netw. 2021. Vol. 122. P.102591. https://doi.org/10.1016/j.adhoc.2021.102591.

[22] ***Liu Z., Fang Y., Huang C., Xu Y.*** MFXSS: An effective XSS vulnerability detection method in JavaScript based on multi-feature model // Comput. Secur. 2023. Vol. 124. P.103015. https://doi.org/10.1016/j.cose.2022.103015.

[23] ***Tian J., Xing W., Li Z.*** BVDetector: A program slice-based binary code vulnerability intelligent detection system // Inf. Softw. Technol. 2020. Vol. 123. P.106289. https://doi.org/10.1016/j.infsof.2020.106289.

[24] ***Kalouptsoglou I., Siavvas M., Ampatzoglou A., Kehagias D., Chatzigeorgiou A.*** Vulnerability classification on source code using text mining and deep learning techniques // In:. Proc. of the 24th IEEE Int. Conf. on Software Quality, Reliability, and Security (QRS'24). 2024. P.1–10.

[25] ***Russo E.R., Di Sorbo A., Visaggio C.A., Canfora G.*** Summarizing vulnerabilities' descriptions to support experts during vulnerability assessment activities // J. Syst. Softw. 2019. Vol. 156. P.84–99. https://doi.org/10.1016/j.jss.2019.06.001.

[26] ***Kühn P., Relke D.N., Reuter C.*** Common vulnerability scoring system prediction based on open source intelligence information sources // Comput. Secur. 2023. Vol. 131. P.103286. https://doi.org/10.1016/j.cose.2023.103286.

[27] ***Котенко И.В., Дойникова Е.В.*** Методы оценивания уязвимостей: использование для анализа защищенности компьютерных систем // Защита информации. Инсайд. № 4 (40). 2011. С.74–81.

[28] ***Feng Z., Guo D., Kim J., Liu S.*** CodeBERT: A Pre-Trained Model for Programming and Natural Languages // Advances in Natural Language Processing. 2020. P.1–12. https://arxiv.org/abs/2002.08155.

[29] ***Devlin J., Chang M.-W., Lee K., Toutanova K.*** BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Journal of Machine Learning Research. 2019. P.1–10. https://arxiv.org/abs/1810.04805.

[30] ***Loshchilov I., Hutter F.*** Decoupled Weight Decay Regularization // In: Proc. of the Int. Conf. on Learning Representations. 2017. P.1–18. https://arxiv.org/abs/1711.05101.

Сведения об авторах

Бусько Никита Алексеевич, 2002 г. рождения. Окончил бакалавриат Санкт-Петербургского государственного электротехнического университета «ЛЭТИ» им. В.И. Ульянова (Ленина) в 2024 г. *nikuto1@mail.ru*.

Федорченко Елена Владимировна, 1986 г. рождения. Окончила Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина) в 2009 г., к.т.н. (2017). Старший научный сотрудник Лаборатории проблем компьютерной безопасности Санкт-Петербургского Федерального исследовательского центра РАН. Является автором более 100 научных публикаций, включая 2 монографии, и двух патентов на изобретения. ORCID: 0000-0001-6707-9153; Author ID (РИНЦ): 641554; Author ID (Scopus): 37097097100. *doynikova@comsec.spb.ru.*

Котенко Игорь Витальевич, 1961 г. рождения. С отличием окончил Военный инженерный Краснознамённый институт им. А.Ф. Можайского в 1983 г. и Военную академию связи в 1987 г., д.т.н. (1999), профессор (2021), заслуженный деятель науки Российской федерации (2023). Главный научный сотрудник и руководитель Лаборатории проблем компьютерной безопасности Санкт-Петербургского Федерального исследовательского центра РАН, профессор Университета ИТМО, СПбГУТ, УрФУ, Харбинского политехнического университета (КНР) и Хэйлунцзянского университета (КНР). В списке научных трудов более 800 работ в области безопасности компьютерных сетей, искусственного интеллекта, телекоммуникационных систем, включая 25 монографий, и более 90 патентов на изобретения и зарегистрированных программ. ORCID: 0000-0001-6859-7120; Author ID (РИНЦ): 110102; Author ID (Scopus): 15925268000. *ivkote@comsec.spb.ru.* ✉.

Поступила в редакцию 01.07.2024, после рецензирования 11.07.2024. Принята к публикации 16.07.2024.

Scientific article DOI: 10.18287/2223-9537-2024-14-3-408-420

Automatic exploit assessment based on deep learning methods

© 2024, N.A. Busko[1], E.V. Fedorchenko[2], I.V. Kotenko[2]✉

[1] *Saint Petersburg Electrotechnical University "LETI", St. Petersburg, Russia*

[2] *St. Petersburg Federal Research Center of the Russian Academy of Sciences, St. Petersburg, Russia*

Abstract

Assessing and prioritizing programs that exploit software vulnerabilities and are used to carry out cyberattacks on a computer system (exploits) is crucial for effectively responding to cyberattacks. This paper presents a method for automatically assessing exploits, where a model is trained to classify exploits using deep learning methods during the design stage, and the trained model is used to derive an assessment of the exploit's criticality during the operation stage. The methodology is based on the hypothesis that the complexity of using an exploit, its consequences, and its assessment depend on the vulnerability being exploited and the source code of the exploit. This proposed methodology differs from existing ones by using the CodeBERT model to classify exploits based on a marked set of exploit source codes and by marking exploit source codes according to the assessments of associated vulnerabilities using the CVSS (Common Vulnerability Scoring System) version 2.0 vulnerability assessment system. For the experimental evaluation of the developed methodology, data sources (Exploits-DB exploit database and NVD vulnerability database) and initial data for experiments were identified, statistical analysis was performed, and an experimental assessment of the accuracy of exploit classification was carried out. The results obtained can be used in designing automatic exploit assessment systems as part of measures to monitor and improve the security of information systems.

Keywords**:** *exploit, vulnerability, assessment, deep learning, data classification, CodeBert*.

For citation: *Busko NA, Fedorchenko EV, Kotenko IV.* Automatic exploit assessment based on deep learning methods [In Russian]. *Ontology of designing*. 2024; 14(3): 408-420. DOI:10.18287/2223-9537-2024-14-3-408-420.

Financial Support: This research was supported by the grant of RSF #23-21-00498 in SPC RAS, https://rscf.ru/en/project/23-21-00498.

Conflict of interest: The authors declare no conflict of interest.

List of figures and tables

References

[1] ***Kotenko IV, Saenko IB, Polubelova OV, Chechulin AA.***Technologies for managing information and security events to protect computer networks [In Russian]. *Information security problems. Computer systems*. 2012; 2: 57-68.

[2] ***Kotenko IV, Ushakov IA.*** Big data technologies for computer security monitoring [In Russian]. *Information Security. Inside*. 2017; 3 (75): 23-33.

[3] ***Mell P, Scarfone K, Romanosky S.*** A complete guide to the Common Vulnerability Scoring System. Version 2.0. Source: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51198.

[4] ***Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Polosukhin I, Kaiser Ł.*** Attention is all you need. In: Neural Information Processing Systems, proc. of the 31st Int. Conf. (NIPS'17). Curran Associates Inc., Red Hook, NY, USA; 2017: 6000–6010.

[5] ***Akram Z, Majid M, Habib S.*** A systematic literature review: usage of logistic regression for malware detection. In: Innovative Computing, proc. of the 2021 Int. Conf. (Lahore, Pakistan, 2021). 2021: 1-8. DOI: 10.1109/ICIC53490.2021.9693035.

[6] ***Ghanem K, Aparicio-Navarro FJ, Kyriakopoulos KG, Lambotharan S, Chambers JA.*** Support Vector Machine for network intrusion and cyber-attack detection. In: Sensor Signal Processing for Defence, proc. of the 2017 Conf. (London, UK, 2017). 2017: 1-5. DOI: 10.1109/SSPD.2017.8233268.

[7] ***Breiman, L***. Bagging predictors. Mach. Learn. 1996; 24: 123–140.

[8] ***Vu QH, Ruta D, Cen L.*** Gradient boosting decision trees for cyber security threats detection based on network events logs // In: Proc. of the 2019 IEEE Int. Conf. on Big Data (Los Angeles, CA, USA, 2019). 2019. P.5921-5928. DOI: 10.1109/BigData47090.2019.9006061.

[9] ***Choubisa M, Doshi R, Khatri N, Kant Hiran K.*** A simple and robust approach of random forest for intrusion detection system in cyber security. In: IoT and Blockchain Technology, proc. of the 2022 Int. Conf. (Ranchi, India, 2022). 2022: 1-5. DOI: 10.1109/ICIBT52874.2022.9807766.

[10] ***Mukherjee S, Sharma N.*** Intrusion detection using naive Bayes classifier with feature reduction. Procedia Technology. 2012; 4: 119–128. Source: https://doi.org/https://doi.org/10.1016/j.protcy.2012.05.017. 455.

[11] ***Prajoy P, Subrato B, Rubaiyat MM, Pinto P, Utku K.*** Artificial Neural Network for Cybersecurity: A Comprehensive Review. Journal of Information Assurance & Security. 2021; 16(1): 10.

[12] ***Krizhevsky A, Sutskever I, Hinton GE.*** ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems. 2012; 15–30. Source: https://arxiv.org/abs/1207.0580.

[13] ***Bengio Y, Ducharme R, Vincent P, Jauvin, C.*** A Neural Probabilistic Language Model. Journal of Machine Learning Research. 1994; 1–18. Source: https://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf.

[14] ***Pascanu R, Mikolov T, Bengio Y.*** On the difficulty of training recurrent neural networks. In: Machine Learning, proc. of the 30th Int. Conf. 2013: 1–12. Source: https://arxiv.org/abs/1211.5063.

[15] ***Laghrissi F, Douzi S, Douzi K, Hssina B.*** Intrusion detection systems using long short-term memory (LSTM). J Big Data. 2021; 8(65). Source: https://doi.org/10.1186/s40537-021-00448-4.

[16] ***Shaul Z, Holzem C.*** Machine learning based source code classification using syntax oriented features. Source: *ArXiv* abs/1703.07638. 2017.

[17] ***Mateless R, Tsur O, Moskovitch R.*** Pkg2Vec: Hierarchical package embedding for code authorship attribution. Future Gener. Comput. Syst. 2021; 116: 49–60. Source: https://doi.org/10.1016/j.future.2020.10.020.

[18] ***Sebastio S, Baranov E, Biondi F, Decourbe O, Given-Wilson T, Legay A, Puodzius C, Quilbeuf J.*** Optimizing symbolic execution for malware behavior classification. Comput. Secur. 2020; 93: 101775. Source: https://doi.org/10.1016/j.cose.2020.101775.

[19] ***Naeem H, Ullah F, Naeem MR, Khalid S, Vasan D, Jabbar S, Saeed S.*** Malware detection in industrial internet of things based on hybrid image visualization and deep learning model. Ad Hoc Netw. 2020; 105: 102154. Source: https://doi.org/10.1016/j.adhoc.2020.102154.

[20] ***Sun H, Cui L, Li L, Ding Z, Hao Z, Cui J, Liu P.*** VDSimilar: Vulnerability detection based on code similarity of vulnerabilities and patches. Comput. Secur. 2021; 110: 102417. Source: https://doi.org/10.1016/j.cose.2021.102417.

[21] ***Moti Z, Hashemi S, Karimipour H, Dehghantanha A, Jahromi AN, Abdi L, Alavi F.*** Generative adversarial network to detect unseen Internet of Things malware. Ad Hoc Netw. 2021; 122: 102591. Source: https://doi.org/10.1016/j.adhoc.2021.102591.

[22] ***Liu Z, Fang Y, Huang C, Xu Y.*** MFXSS: An effective XSS vulnerability detection method in JavaScript based on multi-feature model. Comput. Secur. 2023; 124: 103015. Source: https://doi.org/10.1016/j.cose.2022.103015.

[23] ***Tian J, Xing W, Li Z.*** BVDetector: A program slice-based binary code vulnerability intelligent detection system. Inf. Softw. Technol. 2020; 123: 106289. Source: https://doi.org/10.1016/j.infsof.2020.106289.

[24] ***Kalouptsoglou I, Siavvas M, Ampatzoglou A, Kehagias D, Chatzigeorgiou A.*** Vulnerability classification on source code using text mining and deep learning techniques. In: Software Quality, Reliability, and Security, proc. of the 24th IEEE Int. Conf. (QRS'24). 2024; 1–10.

[25] ***Russo ER, Di Sorbo A, Visaggio CA, Canfora G.*** Summarizing vulnerabilities' descriptions to support experts during vulnerability assessment activities. J. Syst. Softw. 2019; 156: 84–99. Source: https://doi.org/10.1016/j.jss.2019.06.001.

[26] ***Kühn P, Relke DN, Reuter C.*** Common vulnerability scoring system prediction based on open source intelligence information sources. Comput. Secur. 2023; 131: 103286. Source: https://doi.org/10.1016/j.cose.2023.103286.

[27] ***Kotenko I, Doynikova E.*** Vulnerabilities assessment techniques: application in computer systems security analysis [In Russian]. *Information Security*. Inside. 2011; 49: 74-81.

[28] ***Feng Z, Guo D, Kim J, Liu S.*** CodeBERT: A Pre-Trained Model for Programming and Natural Languages. Advances in Natural Language Processing. 2020; 1–12. Source: https://arxiv.org/abs/2002.08155.

[29] ***Devlin J, Chang M-W, Lee K, Toutanova K.*** BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Journal of Machine Learning Research*. 2019; 1–10. Source: https://arxiv.org/abs/1810.04805.

[30] ***Loshchilov I, Hutter F.*** Decoupled Weight Decay Regularization. In: Learning Representations, proc. of the Int. Conf. 2017; 1–18. Source: https://arxiv.org/abs/1711.05101.

About the authors

Nikita Alexeevich Busko (b. 2002) graduated from the Saint Petersburg Electrotechnical University "LETI" in 2024. *nikuto1@mail.ru*.

Elena Vladimirovna Fedorchenko (b. 1986) graduated from the Saint Petersburg Electrotechnical University "LETI" in 2009, PhD (2017). She is a senior researcher at the Laboratory of Computer Security Problems of the St. Petersburg Federal Research Center of the Russian Academy of Sciences. She is a co-author of more than 100 scientific articles including 2 monographs and 2 patents. ORCID: 0000-0001-6707-9153; Author ID (RSCI): 641554; Author ID (Scopus): 37097097100. *doynikova@comsec.spb.ru.*

Igor Vitalievich Kotenko (b. 1961) graduated with honors from the St. Petersburg Academy of Space Engineering in 1983 and the St. Petersburg Signal Academy in 1987, D. Sc. Eng. (1999), professor (2021), Honored Scientist of the Russian Federation (2023). Chief Researcher and Head of the Laboratory of Computer Security Problems at the St. Petersburg Federal Research Center of the Russian Academy of Sciences, professor at ITMO University, Saint Petersburg State University of Telecommunications, Ural Federal University, Harbin Polytechnic University (China) and Heilongjiang University (China). He is a co-author of more than 800 publications in the field of computer network security, artificial intelligence, telecommunication systems, including 25 monographs, and more than 90 patents for inventions and registered programs. ORCID: 0000-0001-6859-7120; Author ID (RSCI): 110102; Author ID (Scopus): 15925268000. *ivkote@comsec.spb.ru.* ✉.

Received July 1, 2024. Revised July 11, 2024. Accepted July 16, 2024.