УДК 004.89

А.В. Куртукова

Интегрированный подход к идентификации вредоносных программ на основе динамического анализа и глубокого обучения

Представлен новый подход к идентификации вредоносных программ. В его основе лежит идея интеграции методов анализа поведения программ с современными алгоритмами машинного обучения. Процесс включает дизассемблирование программ, построение графа потока управления, выявление поведенческих паттернов в изолированной среде, извлечение метаинформации и классификацию программ по 3 классам. Алгоритмической основой разработанного подхода является ансамбль из графовой и гибридной нейронных сетей. Целью графовой сети является анализ графа потока управления, а гибридной – анализ статических и динамических признаков, определенных Cockoo Sandbox, а также ассемблерного кода, полученного в результате реверс-инжиниринга. Подход на базе такого ансамбля демонстрирует точность 0,88 в классификации кода на легитимный, вредоносный и APT-вредоносный и 0,94 – на легитимный и вредоносный.

Ключевые слова: вредоносный код, APT, статический анализ, динамический анализ, вирус.
DOI: 10.21293/1818-0442-2024-28-1-108-113

С увеличением количества и сложности кибератак актуальным становится вопрос определения авторства вредоносного программного кода. Традиционные методы, такие как статический анализ кода или использование сигнатур, часто оказываются недостаточными для выявления злоумышленников, особенно когда речь идет о группировках авторов целевых продолжительных атак (APT-группировках). Такие группы, как Lazarus, Turla, Kimsuky и пр., используют сложные техники маскировки вредоносных признаков и обфускации программного кода, что делает их идентификацию еще более трудоемкой. Тем не менее программы, разрабатываемые такими APT-группировками, имеют отличительные признаки (архитектуру, технику обхода антивирусной защиты и алгоритмы шифрования), что подчеркивает возможность разграничения легитимного (наиболее вероятно «безопасного»), вредоносного и APT-вредоносного кода.

Таким образом, важной задачей является разработка подхода, позволяющего выполнять углубленный анализ программы на предмет ее вредоносности, с целью его последующего применения в процессе идентификации авторства программного кода.

Вопросу идентификации программы как вредоносной посвящены труды [1–6].

В исследовании [1] представлен алгоритм обнаружения вредоносного исходного кода с использованием трансформера (MSDT). Исследование авторов состоит из 4 этапов. Первый этап – формирование набора данных. Всего на данном этапе было собрано 670 тыс. образцов кода на Python с ресурса GitHub. В собранные образцы были искусственно внедрены вредоносные фрагменты из набора данных «The Backstabber's Knife Collection» (BKC) [2].

Второй этап – векторизация набора данных при помощи энкодера на базе трансформера.

Третьим этапом является обнаружение аномалий при помощи плотностного алгоритма пространственной кластеризации с присутствием шума (DBSCAN): именно этот этап позволяет выявлять инъектированные вредоносные фрагменты. На последнем этапе происходит ранжирование аномалий по расстоянию до ближайших точек кластера. Точность обнаружения вредоносных фрагментов в исходных кодах достигает 0,909 для 20 образцов кода.

Авторы работы [3] представляют подход CodeOntology. В его основе лежит идея моделирования элементов программного кода (классов, интерфейсов, методов и переменных) с помощью онтологической структуры. CodeOntology реализован на основе OWL2 [4], которая позволяет визуализировать взаимосвязи между элементами кода на языке Java. Для преобразования исходного кода в RDF-тройки используется специализированный парсер. RDF-тройки представляют собой структурированные данные, состоящие из трех компонентов: субъекта, предиката и объекта, что позволяет описывать отношения между различными элементами кода. Ключевым аспектом данного метода является выявление сигнатур кода – уникальных характеристик, таких как методы вызовов и доступ к определенным файлам. На основе этих сигнатур создаются классы и объекты в CodeOntology, что позволяет осуществлять поиск уязвимых участков кода. Результаты экспериментов продемонстрировали высокую эффективность данного подхода в обнаружении возможных угроз сетевой безопасности. Численные метрики авторами не приводятся.

В работе [5] рассматривается два ключевых вопроса. Во-первых, насколько эффективны трансформерные (CodeBERT, GraphCodeBERT, RoBERTa) модели для построения векторных представлений кода, во-вторых, насколько точно классификаторы (случайный лес (RF), наивный Байес, метод опорных векторов, перцептрон, *k*-ближайших соседей, DBSCAN), обученные на таких представлениях, способны выявлять вредоносные фрагменты кода. Для формирования экспериментального набора вредоносных образцов из датасета BKC было отобрано 1 402 файла на

языке JavaScript. Набор легитимных образцов получен с помощью запросов npm API4, всего в него вошло 26 067 исходных кодов. Лучший результат достигнут классификатором RF, обученным на CodeBERT/GraphCodeBERT-представлениях, F1 составила 0,98, точность – 1, полнота – 0,98.

Статья [6] посвящена подходу к определению вредоносного кода на основе классификатора графа потока управления (CFG). Авторы предлагают классифицировать векторизованный при помощи мешка слов программный код нейронной сетью (НС) адаптивно-резонансной теории. Для оценки подхода было сгенерировано 2 класса состояний CFG по 6 тыс. образов каждый. Генерация вредоносных образов выполнялась за счет незакрытого сетевого сокета и неосвобожденного файлового дескриптора. Точность определения легитимного кода составила 0,95, вредоносного – 0,97. Разница в точности обусловлена случайными факторами в процессе генерации.

Большинство современных исследований сосредоточено на определении того, содержит ли программный код блоки с признаками вредоносности. Предлагаемые методы представляют ценность для решения задач информационной безопасности: анализа сетевого трафика и выявления вирусного программного обеспечения. Лежащие в основе исследований идеи и отдельные элементы методов могут служить отправной точкой для разработки подхода к идентификации вредоносных программ.

Таким образом, целью данного исследования является разработка нового подхода к идентификации вредоносных программ и программного модуля на его основе.

Новый подход основан на интеграции метода анализа поведения программы с современными алгоритмами машинного обучения и включает несколько этапов:

1) дизассемблирование программы и построение CFG на основе ассемблерного кода;

2) выявление поведенческих паттернов в процессе работы вредоносной программы в изолированной среде (динамических признаков);

3) извлечение метаинформации о программе (статических признаков);

4) определение класса, к которому относится программа, ее классификация как легитимной, вредоносной или APT-вредоносной.

Набор данных

Формирование набора данных для обучения и оценки классификаторов являлось наиболее сложной и важной составляющей исследования. Это обусловлено спецификой данных, которая требует осторожности при работе с ними, и их труднодоступностью.

Существует ряд открытых ресурсов, позволяющих собрать разнообразный набор данных, среди которых можно выделить: Virus Total [7], Malware Bazaar [8], GitHub [9], Kaggle [10] и Cyber Science Lab. [11]. Часть образцов вредоносного кода была получена из датасета BKC.

Информация о сформированном наборе данных представлена в табл. 1.

Таблица 1

Статистика набора данных

Характеристика	Легитимные	Вред.	APT-вред.
Общее число авторов	167	–	12
Общее число файлов	5 932	457	2 821
Число файлов с изв. автором	5 661	–	2 821
Число обфусц. файлов	–	35	–
Макс. число файлов автора	51	–	241
Мин. число файлов автора	20	–	43
Ср. число строк кода на автора	1 677	–	1 987

Несколько важных аспектов:

– легитимные файлы были собраны в рамках предшествующих исследований [12–14];

– недостаток информации об авторстве вредоносных файлов обусловлен тем, что большинство таких программ разрабатываются анонимно;

– для APT-вредоносных программ под автором подразумевается конкретная группа программистов из числа группировок: DarkHotel, Energetic Bear, Equation Group, Gorgon Group, Winnti, BlackTech, Kimsuky, Turla, а также ряд группировок без названия (APT1, APT3, APT28, APT40).

Интегрированный подход к идентификации вредоносных программ на основе динамического анализа и глубокого обучения

Процесс работы программного модуля, основанного на подходе к идентификации вредоносных программ, представлен на рис. 1. Основная идея состоит в использовании для обучения НС непосредственно дизассемблированного кода, CFG, построенного на его основе, а также статических и динамических признаков из отчета, формируемого Cockoo Sandbox [15] по результатам запуска программы в изолированной среде. Представленный процесс состоит из 4 основных этапов.

На этапе 1 происходит процесс сбора и подготовки данных, описанный выше.

На этапе 2 производится запуск исполняемых файлов в изолированной среде выполнения Cockoo Sandbox, развернутой внутри виртуальной машины с операционной системой Linux. В процессе выполнения файла Cockoo Sandbox фиксирует различные аспекты его поведения: системные вызовы, сетевые соединения, изменения в файловой системе и реестре, создание потоков и процессов и пр. Результатом работы Cockoo Sandbox является подробный отчет в json-формате, содержащий статические и динамические характеристики программы. В отдельных случаях Cockoo Sandbox делает собственное заключение о вредоносности за счет интеграции с обширной базой знаний о вредоносном ПО.

На этапе 3 производится дизассемблирование программы при помощи инструмента реверс-инжи-

ниринга IDA Pro [16]. Результатом этого процесса являются ассемблерный код, который возможно анализировать по аналогии с исходными кодами на высокоуровневых языках программирования [12, 13], и CFG – ориентированный граф, узлы которого соответствуют базовым блокам, а ребра – возможным путям выполнения этих блоков.

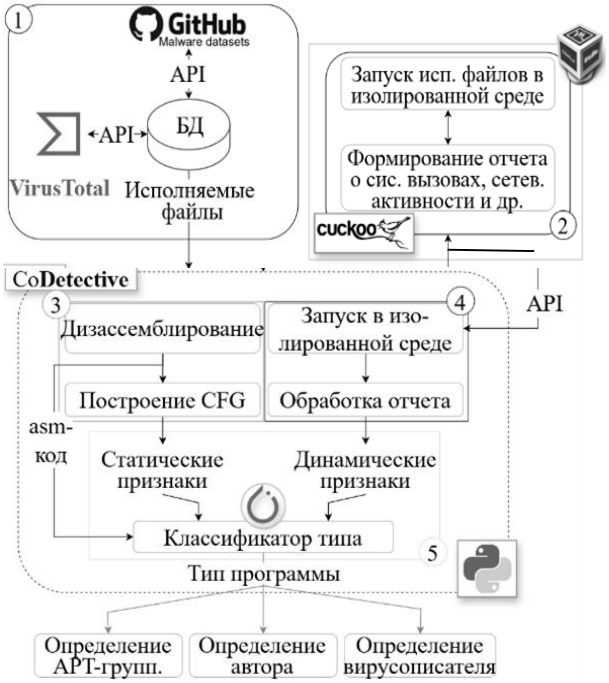

Рис. 1. Модуль идентификации вредоносных программ

На последнем этапе признаки, полученные на 2-м и 3-м этапах, передаются в классификатор, определяющий, является ли код легитимным, вредоносным или АРТ-вредоносным. Исходя из результата работы классификатора, выносится решение о том, какой из ранее разработанных алгоритмов [12–14] будет использоваться для определения автора программы.

Классификация программного кода как легитимного, вредоносного или АРТ-вредоносного

Для обеспечения эффективности работы программного модуля необходимо, чтобы алгоритмическая составляющая демонстрировала высокую точность классификации. Поэтому важной частью исследования являлся выбор архитектуры, способной выполнять точный анализ статических и динамических признаков, полученных от IDA Pro и Cockoo Sandbox. В качестве таких архитектур рассмотрены:

– GNN – графовая НС для эффективной обработки CFG, реализованная с помощью библиотеки PyTorch Geometrics [17];

– HNN – гибрид архитектур Inception и двунаправленных управляемых рекуррентных блоков [13];

– BERT с классификационным слоем – трансформерная модель, обученная на естественно-языковых текстах на различных языках;

– CodeBERT с классификационным слоем – трансформерная модель, обученная на естественно-языковых текстах и исходных кодах программ на различных языках;

– новый ансамбль GNN (2 последовательных GCNConv слоя + линейный слой) и HNN с классификационным слоем, объединяющим выходы обеих НС (процесс обучения представлен UML-диаграммой активности на рис. 2).

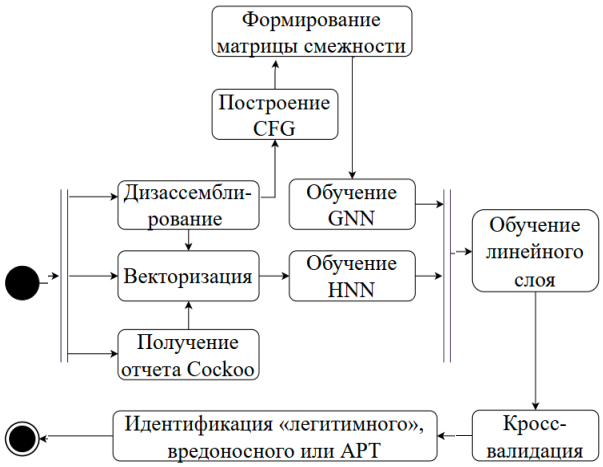

Рис. 2. UML-диаграмма обучения ансамбля

В рамках эксперимента оценивалась эффективность классификации для 3 случаев:

1. 3 класса: легитимный, вредоносный и АРТ.
2. 2 класса: легитимный и объединенные в общий класс вредоносный и АРТ.
3. 2 класса: вредоносный и АРТ.

Метрики точности, полноты, правильности и F1 по результатам процедуры кросс-валидации по 10-фолдам представлены в табл. 2.

Таблица 2

Результаты экспериментов

Модель	Метрика	Лег. – вред. – АРТ-вред.	Лег. – вред и – АРТ-вред.	Вред. – АРТ-вред.
GNN	Точность	0,85±0,02	0,915±0,02	0,69±0,01
	Полнота	0,86±0,01	0,92±0,02	0,69±0,02
	Правил.	0,85±0,02	0,91±0,01	0,7±0,04
	F1	0,85±0,01	0,92±0,01	0,7±0,015
HNN	Точность	0,845±0,03	0,93±0,02	0,65±0,02
	Полнота	0,85±0,01	0,94±0,04	0,67±0,03
	Правил.	0,84±0,015	0,93±0,02	0,65±0,01
	F1	0,845±0,01	0,94±0,02	0,66±0,03
BERT	Точность	0,77±0,02	0,88±0,03	0,635±0,02
	Полнота	0,785±0,02	0,88±0,01	0,85±0,04
	Правил.	0,78±0,04	0,88±0,03	0,83±0,03
	F1	0,78±0,03	0,88±0,02	0,83±0,03
CodeBERT	Точность	0,865±0,01	0,93±0,02	0,73±0,02
	Полнота	0,86±0,05	0,935±0,02	0,75±0,03
	Правил.	0,87±0,03	0,93±0,03	0,73±0,01
	F1	0,87±0,04	0,935±0,02	0,735±0,02
GNN+ HNN	Точность	**0,88±0,01**	**0,94±0,02**	**0,745±0,02**
	Полнота	**0,9±0,02**	**0,96±0,04**	**0,75±0,03**
	Правил.	**0,88±0,02**	**0,95±0,05**	**0,75±0,02**
	F1	**0,89±0,02**	**0,95±0,04**	**0,75±0,03**

Для всех 3 рассмотренных случаев были проведены тесты, направленные на определение статистической значимости полученных результатов точности. Непараметрический тест Фридмана был направлен на выявление общих различий между результатами моделей на основе измерений, полученных на каждом из фолдов кросс-валидации. В случае, если тест Фридмана указывал на значимые различия, применялся тест Неменьи. С его помощью выполнялось попарное сравнение всех моделей, чтобы выявить, между какими моделями наблюдаются статистически значимые различия. Так как для всех 3 случаев нулевая гипотеза (различия результатов статистически незначимы) была отвергнута, была проведена серия тестов Неменьи. Результаты для всех случаев классификаций представлены на рис. 3–5. Значения, выделенные непрерывной линией, – *p*-значения в диапазоне от 0,7 до 1, пунктирной линией – от 0,35 до 0,7, другие – от 0 до 0, 35.

	GNN	HNN	BERT	Code-BERT	GNN+HNN
GNN	1,0	0,436	0,002	0,999	0,618
HNN	0,436	1,0	0,276	0,618	0,016
BERT	0,002	0,276	1,0	0,006	0,000
Code-BERT	0,999	0,618	0,006	1,0	0,436
GNN+HNN	0,618	0,016	0,000	0,436	1,0

Рис. 3. Классификация на 3 класса

	GNN	HNN	BERT	Code-BERT	GNN+HNN
GNN	1,0	1,0	0,081	0,980	0,860
HNN	1,0	1,0	0,081	0,980	0,860
BERT	0,081	0,081	1,0	0,276	0,004
Code-BERT	0,980	0,980	0,276	1,0	0,526
GNN+HNN	0,860	0,860	0,004	0,526	1,0

Рис. 4. Классификация на легитимные и вредоносные

Лучшие метрики во всех случаях получены ансамблем моделей GNN, обученной на CFG, и HNN, обученной на дизассемблированных кодах и признаках, извлеченных из отчета Cockoo Sandbox. Такой результат обусловлен высокой эффективностью каждой из архитектур применительно к отдельно взятым признакам. Второй по эффективности результат демонстрирует CodeBERT. Несмотря на конкурентную эффективность CodeBERT, важно отметить, что она имеет в 10 раз больше параметров, чем ансамбль GNN+HNN, и требует больших вычислительных и временных ресурсов для обучения.

	GNN	HNN	BERT	Code-BERT	GNN+HNN
GNN	1,0	0,211	0,010	0,860	0,526
HNN	0,211	1,0	0,790	0,016	0,002
BERT	0,010	0,790	1,0	0,000	0,000
Code-BERT	0,860	0,016	0,000	1,0	0,980
GNN+HNN	0,526	0,002	0,000	0,980	1,0

Рис. 5. Классификация на вредоносные и APT-вредоносные

Наибольшую сложность для всех моделей представляет случай с разграничением вредоносных и APT-вредоносных программ. Это связано с тем, что такие программы демонстрируют схожие поведенческие паттерны в изолированной среде. Полученная точность 0,745 в дальнейшем может быть повышена за счет аугментации набора данных вредоносных программ.

Обученный ансамбль GNN+HNN с классификационным слоем стал алгоритмической основой программного модуля идентификации вредоносного программного кода (см. рис. 1), входящего в состав программного комплекса «CoDetective» [18].

Заключение

В рамках данного исследования были разработаны подход и программный модуль для идентификации вредоносного программного кода. Подход позволяет идентифицировать программный код как легитимный, вредоносный или APT-вредоносный.

В основе разработанного подхода лежит интеграция динамического анализа поведения программы и методов глубокого обучения. Динамический анализ реализуется за счет использования сведений из отче-

тов Cockoo Sandbox в то время, как глубокие НС обучаются на данных, полученных в результате реверс-инжиниринга – ассемблерном коде и CFG. В рамках проведенных экспериментов доказана особая эффективность ансамблевой архитектуры, сочетающей GNN и HNN. Такая модель позволяет добиться точности 0,88 для 3-классовой классификации, 0,94 – для случая разграничения легитимного и вредоносного кода и 0,745 – для случая разграничения вредоносного и APT-вредоносного кода.

В продолжение данного исследования будет выполнено внедрение подхода к идентификации вредоносных программ и программного модуля на его основе в программный комплекс CoDetective. Усовершенствованный программный комплекс будет апробирован для решения задач идентификации автора-программиста, вирусописателя или APT-группировки, создавшего или создавшей программный код.

Исследование выполнено при финансовой поддержке Министерства науки и высшего образования РФ в рамках базовой части государственного задания ТУСУРа на 2023–2025 гг. (проект № FEWM-2023-0015).

Литература

1. Tsfaty C. Malicious Source Code Detection Using Transformer / C. Tsfaty, M. Fire // arXiv preprint. – arXiv: 2209.07957. – 2022. – URL: https://arxiv.org/abs/2209.07957, свободный (дата обращения: 09.02.2025).

2. The Backstabber's Knife Collection [Электронный ресурс]. – URL: https://dasfreak.github.io/Backstabbers-Knife-Collection, свободный (дата обращения: 09.02.2025).

3. Navid S.Z. Static Detection of Malicious Code in Programs Using Semantic Techniques / S.Z. Navid, P. Dey, S. Hasan, M. Ali // 2020 11th International Conference on Electrical and Computer Engineering (ICECE). – 2020. – P. 327–330. DOI: 10.1109/ICECE51571.2020.9393121.

4. OWL2. – URL: https://www.w3.org/TR/owl2-overview, свободный (дата обращения: 09.02.2025).

5. Using Pre-trained Transformers to Detect Malicious Source Code Within JavaScript Packages / M. Klein, D. Krupka, C. Winter, M. Gergeleit, L. Martin // Informatik. Lecture Notes in Informatics (LNI). – 2024. – P. 529–538. DOI: 10.18420/inf2024.

6. Буханов Д.Г. Выявление вредоносного программного обеспечения на основе классификации графов потоков исходных кодов / Д.Г. Буханов, Д.В. Сулохин // Доклады ТУСУР. – 2018. – Т. 21, № 3. – С. 30–34. DOI: 10.21293/1818-0442-2018-21-3-30-34.

7. Virus Total [Электронный ресурс]. – URL: https://www.virustotal.com, свободный (дата обращения: 09.02.2025).

8. Malware Bazaar. – URL: https:// bazaar.abuse.ch, свободный (дата обращения: 09.02.2025).

9. SourceFinder: Finding Malware Source-Code from Publicly Available Repositories / O.F. Rokon, R. Islam, A. Darki, E.E. Papalexakis, M. Faloutsos // arXiv preprint. – arXiv: 2005.14311. – 2020. – URL: https://arxiv.org/abs/2005.14311, свободный (дата обращения: 09.02.2025).

10. Kaggle. – URL: https://www.kaggle.com/, свободный (дата обращения: 09.02.2025).

11. Cyber Science Lab. APT Malware dataset. – URL: https://cybersciencelab.com/advanced-persistent-threat-apt-malware-dataset/, свободный (дата обращения: 09.02.2025).

12. Kurtukova A. Source Code Authorship Identification Using Deep Neural Networks / A. Kurtukova, A. Romanov, A. Shelupanov // Symmetry. – 2020. – Vol. 12. – P. 2044. – DOI:10.3390/sym12122044.

13. Complex Cases of Source Code Authorship Identification Using a Hybrid Deep Neural Network / A. Kurtukova, A. Romanov, A. Shelupanov, A. Fedotova // Future Internet. – 2022. – Vol. 14. – P. 287. DOI: 10.3390/fi14100287.

14. Куртукова А.В. Разработка методики идентификации авторства бинарных и дизассемблированных кодов программы на основе ансамбля современных методов обработки естественного языка / А.В. Куртукова, А.С. Романов, А.А. Шелупанов // Доклады ТУСУР. – 2023. – Т. 26, № 4. – С. 53–60. DOI: 10.21293/1818-0442-2023-26-4-53-60.

15. Cockoo Sandbox [Электронный ресурс]. – URL: https://cuckoosandbox. org/index.html, свободный (дата обращения: 09.02.2025).

16. IDA Pro [Электронный ресурс]. – URL: https://hex-rays.com/ida-pro, свободный (дата обращения: 09.02.2025).

17. PyTorch Geometrics [Электронный ресурс]. – URL: https://pytorch-geometric.readthedocs.io, свободный (дата обращения: 09.02.2025).

18. Св-во о гос. регистр. программы для ЭВМ № 2021681140. Система для идентификации автора исходного кода программы «CoDEtective» / А.В. Куртукова, А.С. Романов. – Заявка №2021667210. Дата поступления: 26 октября 2021. Зар. в Реестре программ для ЭВМ 17 декабря 2021 г.

Куртукова Анна Владимировна
Аспирант каф. комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС) Томского государственного университета систем управления и радиоэлектроники (ТУСУР)
Ленина пр-т, 40, г. Томск, Россия, 634050
Тел.: +7-905-991-67-13
Эл. почта: av.kurtukova@gmail.com

Поступила в редакцию: 14.02.2025.
Принята к публикации: 04.04.2025.

Kurtukova A.V.

An Integrated Approach to Malware Identification Based on Dynamic Analysis and Deep Learning

The article presents a new approach to malware identification. It is based on the idea of integrating program behavior analysis methods with modern machine learning algorithms. The process includes program disassembly, control flow graph construction, behavioral patterns detection in an isolated environment, metainformation extraction and program classification into 3 classes. The algorithmic basis of the developed approach is an ensemble of graph and hybrid neural networks. The purpose of the graph network is to analyze the control flow graph, and the hybrid network is to analyze static and dynamic features defined by Cockoo Sandbox, as well as assembly code obtained as a result of reverse engineering. The approach based on such an ensemble demonstrates an accuracy of 0.88 in classifying code into legitimate, malicious and APT malware and 0.94 - into legitimate and malicious.

Keywords: malware, APT, static analysis, dynamic analysis, virus.
DOI: 10.21293/1818-0442-2024-28-1-108-113

References

1. Tsfaty C., Fire M. Malicious Source Code Detection Using Transformer. *arXiv preprint*, arXiv: 2209.07957, 2022. Available at: https://arxiv.org/abs/2209.07957, free (Accessed: February 09, 2025)

2. The Backstabber's Knife Collection. Available at: https://dasfreak.github.io/Backstabbers-Knife-Collection, free (Accessed: February 09, 2025).

3. Navid S.Z., Dey P., Hasan S., Ali M. Static Detection of Malicious Code in Programs Using Semantic Techniques. *2020 11th International Conference on Electrical and Computer Engineering (ICECE)*, 2020, pp. 327–330. DOI: 10.1109/ICECE 51571.2020.9393121.

4. OWL2. Available at: https://www.w3.org/TR/owl2-overview, free (Accessed: February 09, 2025).

5. Klein M., Krupka D., Winter C., Gergeleit M., Martin L. Using Pre-trained Transformers to Detect Malicious Source Code Within JavaScript Packages. *Informatik,* Lecture Notes in Informatics (LNI), 2024, pp. 529–538. DOI: 10.18420/inf2024.

6. Bukhanov D.G., Sulokhin D.V. [Detection of malware based on the classification of source code graphs]. *Proceedings of TUSUR University,* 2018, vol. 21 no. 3, pp. 30–34. DOI: 10.21293/1818-0442-2018-21-3-30-34 (in Russ.).

7. Virus Total. Available at: https:// www.virustotal.com, free (Accessed: February 09, 2025).

8. Malware Bazaar. Available at: https:// bazaar.abuse.ch, free (Accessed: February 09, 2025).

9. Rokon O.F., Islam R., Darki A., Papalexakis E.E., Faloutsos M. SourceFinder: Finding Malware Source-Code from Publicly Available Repositories. *arXiv preprint*, arXiv: 2005.14311, 2020. Available at: https://arxiv.org/abs/2005. 14311 (Accessed: February 09, 2025).

10. Kaggle. Available at: https://www.kaggle.com, free (Accessed: February 09, 2025).

11. Cyber Science Lab. APT Malware dataset. Available at: https://cybersciencelab.com/advanced-persistent-threat-apt-malware-dataset, free (Accessed: February 09, 2025).

12. Kurtukova A., Romanov A., Shelupanov A. Source Code Authorship Identification Using Deep Neural Networks. *Symmetry*, 2020, vol. 12, 2044. DOI:10.3390/sym12122044.

13. Kurtukova A., Romanov A., Fedotova A., Shelupanov A. Complex Cases of Source Code Authorship Identification Using a Hybrid Deep Neural Network. *Future Internet,* 2022, vol. 14, 287. DOI: 10.3390/fi14100287.

14. Kurtukova A., Romanov A., Shelupanov A. [Development of a methodology for identifying the authorship of binary and disassembled program codes based on an ensemble of modern natural language processing methods]. *Proceedings of TUSUR University,* 2023, vol. 26, no. 4, pp. 53–60. DOI: 10.21293/1818-0442-2023-26-4-53-60. (In Russ.).

15. Cockoo Sandbox. Available at: https://cuckoosandbox.org/index.html, free (Accessed: February 09, 2025).

16. IDA Pro. Available at: https://hex-rays.com/ida-pro, free (Accessed: February 09, 2025).

17. PyTorch Geometrics. Available at: https://pytorch-geometric.readthedocs.io, free (Accessed: February 09, 2025).

18. Kurtukova A.V., Romanov A.S. *Sistema dlya identifikacii avtora iskhodnogo koda programmy «CoDEtective»* [System for identifying the author of the source code of the program «CoDetective»]. Rosreestr RF, no. 2021667210, 2021.

Anna V. Kurtukova
Postgraduate student, Department of Complex Information Security of Electronic Computer Systems (KIBEVS),
Tomsk State University of Control Systems and Radioelectronics (TUSUR)
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7-905-991-67-13
Email: av.kurtukova@gmail.com

Received: 14.02.2025.
Accepted: 04.04.2025.