УДК 004.89

А.В. Куртукова, Е.Е. Сваровская, А.С. Романов

Оценка влияния обфускации на процесс идентификации автора программного кода

Различные способы запутывания исходного кода могут снизить эффективность применяемых моделей идентификации автора программного кода до случайного угадывания. Данная статья посвящена оценке влияния факта обфускации исходного кода на процесс идентификации автора программы при помощи модели гибридной нейронной сети. В рамках исследования были проведены эксперименты как с интерпретируемыми, так и с компилируемыми языками программирования. Полученные результаты свидетельствуют об устойчивости предложенной ранее модели гибридной нейронной сети к обфускации и возможности ее применения для решения поставленной задачи.

Ключевые слова: автор, исходный код, обфускация, нейронная сеть, машинное обучение.
doi: 10.21293/1818-0442-2020-23-2-50-54

Обфускация исходного кода программы [1–6] представляет собой процесс, направленный на запутывание текста программы, написанной на какомлибо языке программирования. Наиболее распространенной областью применения обфускации является защита программного кода от плагиата, все чаще встречающегося в сфере коммерческой разработки программного обеспечения (ПО) ввиду активного развития информационных технологий. Разработчики ПО выполняют запутывание кода с помощью специальных инструментов, намеренно делая код трудным для понимания. Однако нельзя исключать также и возможность их применения для сокрытия уникальных приемов, привычек и стиля написания кода, а следовательно, для анонимизации автора программы. Так, авторы-вирусописатели нередко прибегают к обфускации именно с этой целью. В таких ситуациях экспертный анализ исходного кода вручную становится невозможным и возникает необходимость в использовании технических средств идентификации автора исходного кода программы, устойчивых к запутыванию исходного кода.

Несмотря на большое количество исследований, посвященных задаче определения автора исходного кода, тема обфускации в них практически не освещена.

Единственная работа, учитывающая влияние обфускации на идентификацию автора исходного кода, написана группой ученых из Дрексельского университета [7]. Для решения задачи классификации они используют подход на основе рекуррентной нейронной сети (НС), случайного леса и TF-IDF представлений. Авторы рассмотрели влияние обфускатора Tiggers [8], предназначенного для языка программирования C, на идентификацию автора исходного кода. Исходная точность определения автора составляла 93,65%, а точность на обфусцированных данных – 58,33%. Результаты позволяют сделать вывод о негативном влиянии обфускации на точность методов определения авторства исходного кода. Это обусловлено тем, что признаки, на основе которых модель принимает решение, теряют свою информативность вследствие переименования, шифрования и других преобразований кода, применяемых для сокрытия авторской личности.

Таким образом, была поставлена цель – оценить влияние обфусцированных исходных кодов на процесс идентификации автора программного кода моделью гибридной НС (HNN), представленной авторами в работах [9, 10].

Экспериментальный набор данных

Обфускация исходного кода является особо востребованной для интерпретируемых языков программирования, то есть для языков, где операторы транслируются и выполняются последовательно друг за другом. Это обусловлено тем, что программы на таких языках представляют собой скрипт-программный сценарий, описывающий последовательность действий, выполняющийся интерпретатором, а не исполняемый машинный код, зачастую сложнее поддающийся анализу, как в случае с компилируемыми языками.

В рамках данного исследования было решено провести эксперимент для обфусцированных исходных кодов, написанных как на интерпретируемых языках (Python, PHP, JavaScropt), так и на компилируемых языках (C, C++). Информация о наборе данных представлена в табл. 1.

Таблица 1

Экспериментальная база

Язык	Количество кодов	Количество авторов	Средняя длина, символов
PHP	17 158	80	374
JavaScript	18 735	69	397
Python	16 783	57	532
C	17 274	62	1 162
C++	12 366	72	988

В качестве обфускаторов были выбраны свободно доступные на хостинге IT-проектов Github [11] реализации, подходящие для автоматизированного запутывания большого количества исходных кодов. Для каждого языка программирования необходимо было использовать отдельный инструмент обфускации, учитывающий его особенности.

Для обфускации исходных кодов, написанных на JavaScript, использовался «JS Obfuscator Tool» [12]. Принцип его действия состоит в лексическом преобразовании, замене имен функций и переменных, удалении комментариев и пробельных символов. Кроме того, выполняется конвертирование строк в шестнадцатеричные последовательности и их дальнейшее кодирование в base64.

Еще один обфускатор, применяемый для JavaScript, – «JS-obfuscator» [13]. Действия, выполняемые им, аналогичны [12]. Помимо этого, он осуществляет запутывание встраиваемого HTML, PHP и другого кода.

В качестве одного из инструментов обфускации исходных кодов на Python применялся «Opy» [14], производящий простую лексическую обфускацию. Он выполняет замену имен функций и переменных на последовательности символов «l» и «1», а также преобразует строки в наборы случайных символов.

Как альтернативный инструмент использовался обфускатор «Pyarmor» [15], осуществляющий запутывание на более высоком, чем [14], уровне. Обфускация происходит в процессе исполнения байт-кода каждого объекта, а очистка локальных переменных происходит сразу по окончании выполнения функции.

Для исходных кодов на языке PHP были использованы лексические обфускаторы «Yakpro-po» [16] и «PHP Obfuscator» [17]. Оба удаляют пробельные символы и комментарии. Помимо этого, [17] осуществляет скремблирование имен переменных, методов, классов и пр., а также замену условных операторов на конструкцию «if … go».

Для языка C++ было решено использовать инструмент «Cpp Guard» [18]. Данный обфускатор удаляет как однострочные, так и многострочные комментарии, дополняет исходный код псевдосложным «мусорным» кодом, не влияющим на вычислительную сложность программы, удаляет пробелы и символы переводов строк. Также он обрабатывает все препроцессорные директивы, используя объявления и однострочные «else» условия, и заменяет строки на их шестнадцатеричное представление.

Для исходных кодов на языке C применялся «AnalyseC» [19], выполняющий простые преобразования в виде замены имен переменных на последовательности случайных символов.

Гибридная нейронная сеть

В рамках данного исследования было решено использовать авторскую модель HNN (рис. 1), отличившуюся особенно высокой точностью для решения проблемы идентификации автора исходного кода. Данная модель представляет собой НС, состоящую из сверточных и рекуррентных слоев, а также слоев прямого распространения. Сверточная часть, реализованная по типу архитектуры Inception-V1 [20], включает в себя фильтры различных размерностей, что позволяет выделять как локальные, так и глобальные информативные признаки из текстовой последовательности. Рекуррентная часть, представленная слоем двунаправленной GRU [21], в свою очередь, осуществляет выявление временных зависимостей, встречающихся в прямом и обратном контекстах.

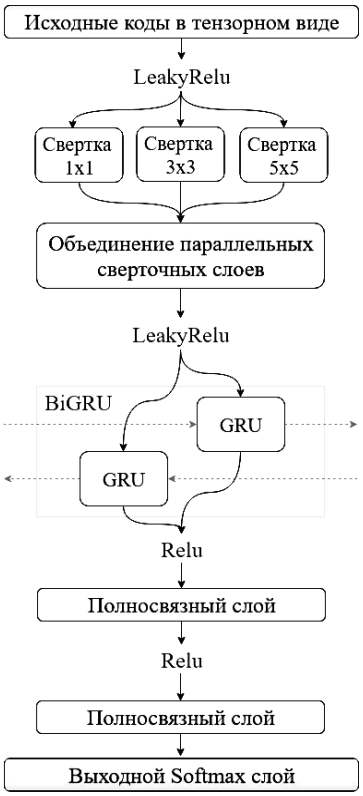

Рис. 1. Модель HNN

Постановка эксперимента и результаты

Существует несколько способов для идентификации автора обфусцированного исходного кода. Один из них – деобфускация, т.е. установление механизма, при помощи которого был запутан исходный код, и его применение для восстановления первоначального вида кода. Другой способ заключается в извлечении устойчивых к запутыванию функций, изменение которых неизбежно повлечет за собой нарушение работоспособности программы. Однако эта задача является нетривиальной. Поиск и выявление таких функций требуют проведения серьезного исследования для каждого из интересующих языков программирования, так как неправильный их выбор может негативно сказаться на процессе идентификации. Наконец, данная задача может быть решена с помощью НС. В частности, глубокие архитектуры могут обучаться на данных независимо от вида их представления и реагировать на признаки, заведомо неизвестные даже эксперту. Подходящей для решения данной задачи является описанная в предыдущем разделе авторская модель HNN.

Для каждого языка программирования были выбраны корпуса, включающие в себя исходные коды, написанные 5, 10 и 20 авторами. Все исходные коды были обфусцированы соответствующими инструментами, описанными в предыдущем разделе. Оценка точности идентификации осуществлялась с помощью перекрестной проверки по 5 блокам.

Результаты проведенных экспериментов представлены в табл. 2 и содержат измерения как до обфускации исходных кодов, так и после.

Таблица 2

Результаты экспериментов с архитектурами

Язык	Обфускация	Количество авторов		
		5	10	20
Java-Script	Без обфускации, %	92	82	76
	JS Obfuscator Tool [12], %	86	80	70
	JS-obfuscator [13], %	86	70	63
Python	Без обфускации, %	95	92	91
	Opy [14], %	87	80	48
	Pyarmor [15], %	70	54	38
PHP	Без обфускации, %	92	89	86
	Yakpro-po [16], %	89	76	63
	PHP Obfuscator [17], %	82	74	61
CPP	Без обфускации, %	92	92	90
	C++ Obfuscator [18]	71	67	41
C	Без обфускации, %	96	95	94
	C Obfuscator [19], %	90	81	76

Результаты экспериментов позволяют сделать вывод об устойчивости авторской модели HNN к лексической обфускации: удалению пробельных символов, преобразованию строк, их конвертированию и кодированию и т.д.

Точность идентификации автора лексически запутанного исходного кода оказывается ниже в среднем на 7%, чем исходного экземпляра. Однако в случае более сложной обфускации, выполняемой такими инструментами, как [15] и [18], где происходит запутывание байт-кода объектов, дополнение исходного кода псевдосложным кодом и др., разность в точности идентификации достигает 30%.

Заключение

В данной статье была произведена оценка влияния обфускации на процесс идентификации автора исходного кода программы при помощи авторской модели HNN.

В рамках исследования были проведены эксперименты с интерпретируемыми (JavaScript, Python, PHP) и компилируемыми (C, C++) языками программирования. С целью запутывания исходных кодов были использованы свободно распространяемые лексические обфускаторы.

Для всех 5 языков точность идентификации автора исходного кода после обфускации оказалась приемлемой и составила в среднем 87%, что оказалось всего на 8% ниже исходной точности.

Таким образом, можно назвать модель HNN устойчивой к простой лексической обфускации, которая может быть выполнена программистом как вручную, так и с использованием свободно распространяемых решений, и применимой при условии небольшого количества предполагаемых авторов наряду с другими инструментами интеллектуального анализа текста [22].

Исследование проводится при поддержке Фонда содействия инновациям по договору № 334ГУЦЭС8-D3/56686 от 27.12.2019 г.

Литература

1. Popa M. Techniques of Program Code Obfuscation for Secure Software // Journal of Mobile, Embedded and Distributed Systems. – 2011. – No. 3. – P. 205–219.

2. Popa M. Characteristics of Program Code Obfuscation for Reverse Engineering of Software // Proceedings of the 4th International Conference on Security for Information Technology and Communications. – 2011. – P. 103–112.

3. The Effectiveness of Source Code Obfuscation: An Experimental Assessment / M. Ceccato., M. Di Penta, J. Nagra, P. Falcarin, F. Ricca, M. Torchiano, P. Tonella // 2009 IEEE 17th International Conference on Program Comprehension. – 2009. – P. 178–187.

4. Anckaert B. Program obfuscation: a quantitative approach / B. Anckaert, M. Madou, B.D. Sutter, B.D. Bus, K.D. Bosschere, B. Preneel // In QoP '07: Proc. of the 2007 ACM Workshop on Quality of protection. – 2007. – P. 15–20.

5. Assessment of Source Code Obfuscation Techniques / A. Viticchié, L. Regano, M. Torchiano, C. Basile, M. Ceccato, P. Tonella, R. Tiella // 2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM). – 2016. – P. 11–20.

6. Буинцев Д.Н. Анализ применения запутывающих преобразований для программного обеспечения / Д.Н. Буинцев, А.А. Шелупанов, О.О. Шевцова // Вопросы защиты информации. – 2005. – № 3. – С. 38–43.

7. Caliskan-Islam A. Deanonymizing programmers via code stylometry / A. Caliskan-Islam, R. Harang, A. Liu // Proceedings of the 24th USENIX Security Symposium. – 2015. – P. 255–270.

8. The tigress diversifying c virtualizer [Электронный ресурс]. – Режим доступа: http://tigress.cs.arizona.edu, свободный (дата обращения: 12.04.2020).

9. Куртукова А.В. Идентификация автора исходного кода методами машинного обучения / А.В. Куртукова, А.С. Романов // Труды СПИИРАН. – 2019. – № 18(3). – С. 741–765.

10. Kurtukova A. De-Anonymization of the Author of the Source Code Using Machine Learning Algorithms / A. Kurtukova. A. Romanov, A. Fedotova // 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). – 2019. – P. 0612–0617.

11. Github [Электронный ресурс]. – Режим доступа: https://github.com/, свободный (дата обращения: 13.04.2019).

12. JS Obfuscator Tool [Электронный ресурс]. – Режим доступа: https://obfuscator.io/, свободный (дата обращения: 26.02.2020).

13. JS-obfuscator [Электронный ресурс]. – Режим доступа: https://github.com/caiguanhao/js-obfuscator, свободный (дата обращения: 26.02.2020).

14. Opy [Электронный ресурс]. – Режим доступа: https://github.com/QQuick/Opy, свободный (дата обращения: 26.02.2020).

15. Pyarmor [Электронный ресурс]. – Режим доступа: https://github.com/dashingsoft/pyarmor, свободный (дата обращения: 26.02.2020).

16. Yakpro-po [Электронный ресурс]. – Режим доступа: https://github.com/pk-fr/yakpro-po, свободный (дата обращения: 26.02.2020).

17. PHP Obfuscator [Электронный ресурс]. – Режим доступа: https://github.com/naneau/php-obfuscator, свободный (дата обращения: 26.02.2020).

18. Cpp Guard [Электронный ресурс]. – Режим доступа: https://github.com/techtocore/Cpp-Guard (дата обращения: 26.02.2020).

19. AnalyseC [Электронный ресурс]. – Режим доступа: https://github.com/ryarnyah/AnalyseC (дата обращения: 26.02.2020).

20. Going deeper with convolutions / C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich // 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2015. – P. 1–9.

21. Mangal S. LSTM vs. GRU vs. Bidirectional RNN for script generation / S. Mangal, J. Poorva, M. Rahul // A Review of Research Topics, Venues, and Top Cited Papers. – 2019. – ArXiv abs/1908.04332.

22. Романов А.С. Обобщенная методика идентификации автора неизвестного текста / А.С. Романов, А.А. Шелупанов, С.С. Бондарчук // Доклады ТУСУР. – 2010. – № 1(21), ч. 1. – С. 108–112.

Куртукова Анна Владимировна
Студентка каф. безопасности информационных систем (БИС) Томского государственного университета систем управления и радиоэлектроники (ТУСУР)
Ленина пр-т, д. 40, г. Томск, 634050
Тел.: +7 (905) 991 6713
Эл. почта: av.kurtukova@gmail.com

Сваровская Елизавета Евгеньевна
Студентка каф. БИС ТУСУРа
Ленина пр-т, д. 40, г. Томск, 634050
Тел.: +7-923-428 88 49
Эл. почта: swarovski991@gmail.com

Романов Александр Сергеевич
Канд. техн. наук, доцент каф. БИС ТУСУРа
Ленина пр-т, д. 40, г. Томск, 634050
Тел.: + 7 (382-2) 41-34-26
Эл. почта: alexx.romanov@gmail.com

Kurtukova A.V., Svarovskaya E.E., Romanov A.S.

Assessing the impact of obfuscation on the process of identifying the author of a program code

Various ways of obfuscating the source code can reduce the effectiveness of the applied models of identifying the author of the program code to random guessing. This article is devoted to assessing the influence of the fact of obfuscation of the source code on the process of identifying the author of a program using a hybrid neural network model. As part of the study, experiments were carried out with both interpreted and compiled programming languages. The results obtained indicate the stability of the previously proposed model of a hybrid neural network to obfuscation and the possibility of its application to solve the problem.

Keywords: author, source code, obfuscation, neural network, machine learning.
doi: 10.21293/1818-0442-2020-23-2-50-54

References

1. Popa M. Techniques of Program Code Obfuscation for Secure Software. *Journal of Mobile, Embedded and Distributed Systems*, no. 3, 2011, pp. 205–219.

2. Popa M. Characteristics of Program Code Obfuscation for Reverse Engineering of Software. *Proceedings of the 4th International Conference on Security for Information Technology and Communications*, 2011, pp. 103–112.

3. Ceccato M., Di Penta M., Nagra J., Falcarin P., Ricca F., Torchiano M., Tonella P. The Effectiveness of Source Code Obfuscation: An Experimental Assessment. 2009 *IEEE 17th International Conference on Program Comprehension*, 2009, pp. 178–187.

4. Anckaert B., Madou M., Sutter B.D., Bus B.D., Bosschere K.D., Preneel B. Program obfuscation: a quantitative approach. *In QoP '*07*: Proc. of the* 2007 *ACM Workshop on Quality of Protection*, 2007, pp. 15–20.

5. Viticchié A., Regano L., Torchiano M., Basile C., Ceccato M., Tonella P., Tiella R. Assessment of Source Code Obfuscation Techniques. 2016 *IEEE* 16*th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2016, pp. 11–20.

6. Buintsev D.N., Shelupanov A.A., Shevtsova O.O. *Analiz primeneniya zaputyvayushchih preobrazovanij dlya programmnogo obespecheniya* [Analysis of the use of obfuscating transformations for software]. Voprosy zashchity informacii [*Information Security Issues*], 2015, vol. 3, pp. 38–43.

7. Caliskan-Islam A., Harang R., Liu A. Deanonymizing programmers via code stylometry. *Proceedings of the* 24*th USENIX Security Symposium*, 2015, pp. 255–270.

8. The tigress diversifying c virtualizer. Available at: http://tigress.cs.arizona.edu (Accessed: April 12, 2020).

9. Kurtukova A.V., Romanov A.S. [Identification author of source code by machine learning methods]. *SPIIRAS Proceedings*, 2019, vol. 18, no. 3. pp. 741–765. (In Russ.).

10. Kurtukova A., Romanov A., Fedotova A. De-Anonymization of the Author of the Source Code Using Machine Learning Algorithms. 2019 *International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, 2019, pp. 0612–0617.

11. Github Available at: https://github.com/ (Accessed: April 12, 2020).

12. JS Obfuscator Tool. Available at: https://obfuscator.io/ (Accessed: April 12, 2020).

13. JS-obfuscator. Available at: https://github.com/caiguanhao/js-obfuscator (Accessed: April 12, 2020).

14. Pyarmor. Available at: https://github.com/dashingsoft/pyarmor (Accessed: April 12, 2020).

15. Opy. Available at: https://github.com/QQuick/Opy (Accessed: April 12, 2020).

16. Yakpro-po. Available at: https://github.com/pkfr/-yakpro-po (Accessed: April 12, 2020).

17. PHP Obfuscator. Available at: https://github.com/-naneau/php-obfuscator (Accessed: April 12, 2020).

18. Cpp Guard. Available at: https://github.com/techtocore/Cpp-Guard (Accessed: April 12, 2020).

19. AnalyseC. Available at: https://github.com/ryarnyah/AnalyseC (Accessed: April 12, 2020).

20. Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. Going deeper with convolutions. 2015 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

21. Mangal S., Poorva J., Rahul M. LSTM vs. GRU vs. Bidirectional RNN for script generation. *A Review of Research Topics, Venues, and Top Cited Papers*, 2019, *ArXiv* abs/1908.04332.

22. Romanov A.S., Shelupanov A.A., Bondarchuk S.S. [Generalized authorship identification technique]. *Proceedings of TUSUR University*, 2010, vol. 1, no. 21, pp. 108–112 (in Russ.).

Anna V. Kurtukova
Student, Department of Information System Security
of 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

Elizaveta E. Svarovskaya
Student, Department of Information System Security TUSUR
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7-923-428-88-49
Email: swarovski991@gmail.com

Aleksandr S. Romanov
Candidate of Engineering Sciences, Associate Professor,
Department of Information System Security, TUSUR
40, Lenin pr., Tomsk, Russia, 634050
Phone: + 7 (382-2) 41-34-26
Email: alexx.romanov@gmail.com