УДК 004.85:004.056.57 DOI: 10.26102/2310-6018/2020.30.3.042

Система обнаружения вредоносного программного обеспечения на основе технологии машинного обучения

О.Н. Выборнова[1], И.А. Пидченко[2]
[1]Астраханский государственный университет
Астрахань, Россия
[2]ООО «Экономатика»
Москва, Россия

Резюме: Непрерывный рост числа вредоносных программ делает актуальной задачу их обнаружения: классификации программ на вредоносные и безопасные. В связи с этим, данное исследование посвящено разработке системы обнаружения вредоносного программного обеспечения на основе машинного обучения, а именно, обучения искусственной нейронной сети с учителем. В ходе исследования проведен анализ структуры исполняемых PE-файлов операционной системы Windows, выбраны характеристики из PE-файлов для формирования обучающего множества, а также выбраны и обоснованы топология (четырехуровневый персептрон) и параметры антивирусной нейронной сети. Для создания и обучения модели использовалась библиотека Keras. При формировании обучающего множества применялась база данных безопасного и вредоносного программного обеспечения Ember. Выполнено обучение и проверка адекватности обучения разработанной модели распознавания вредоносного кода. Результаты обучения предложенной в рамках исследования антивирусной нейронной сети показали высокую точность обнаружения вредоносных программ и отсутствие эффекта переобучения, что свидетельствует о хороших перспективах применения модели. Хотя экспериментальная модель нейронной сети пока не способна полностью заменить антивирусные сканеры, материалы статьи представляют практическую ценность для задач классификации программ на вредоносные и безопасные.

Ключевые слова: вредоносное ПО, машинное обучение, антивирусная нейронная сеть, обучение нейронной сети, Keras, Ember, Dropout.

Для цитирования: Выборнова О.Н., Пидченко И.А. Система обнаружения вредоносного программного обеспечения на основе технологии машинного обучения. *Моделирование, оптимизация и информационные технологии.* 2020;8(3). Доступно по: https://moit.vivt.ru/wp-content/uploads/2020/08/VybornovaPidchenko_3_20_1.pdf DOI: 10.26102/2310-6018/2020.30.3.042

Malware detection system based on machine learning technology

O.N. Vybornova[1], I.A. Pidchenko[2]
[1]Astrakhan State University
Astrakhan, Russia
[2]«Ekonomatika OOO» (limited liability company)
Moscow, Russian Federation

Abstract: The continuous growth in the number of malicious programs makes the task of their detection urgent: classifying programs into malicious and safe. In this regard, this study is devoted to the development of a malware detection system based on machine learning, namely, training an artificial neural network with a teacher. In the course of the study, we analyzed the structure of Portable Executable files of the Windows operating system, selected characteristics from PE-files to form a training set, and also selected and substantiated the topology (four-level perceptron) and parameters of

the antivirus neural network. The Keras library was used to create and train the model. The Ember dataset of safe and malicious software was used to form the training set. We have trained and verified the adequacy of training for the developed malicious code recognition model. The training results of the anti-virus neural network proposed in the study showed a high accuracy of malware detection and the absence of the overtraining effect, which indicates good prospects for using the model. Although the experimental model of a neural network is not able to fully replace the anti-virus scanners, the materials of the article are of practical value for the tasks of classifying programs into malicious and safe.

Keywords: malware, machine learning, anti-virus neural network, neural network training, Keras, Ember, Dropout.

For citation: Vybornova O.N., Pidchenko I.A. Malware detection system based on machine learning technology. *Modeling, Optimization and Information Technology*. 2020;8(3). Available from: https://moit.vivt.ru/wp-content/uploads/2020/08/VybornovaPidchenko_3_20_1.pdf DOI: 10.26102/2310-6018/2020.30.3.042 (In Russ).

Введение (Introduction)

Вредоносные программы представляют серьёзную угрозу как для физических лиц, так и для государственных и коммерческих организаций различных секторов экономики и различных масштабов. Это связано с тем, что атакующие используют вредоносный код для дальнейших сетевых атак и кражи информации. Поэтому своевременное обнаружение таких вторжений и реагирование на них становится решающим для специалистов по кибербезопасности.

Непрерывный рост числа вредоносных программ делает актуальной задачу их обнаружения. В настоящее время эта задача традиционно решается средствами сигнатурного и эвристического анализа. На Рисунках 1 и 2 продемонстрирована статистика обнаружения вредоносного ПО средствами Антивируса Касперского [1].

Рисунок 1 – Количество обнаружений
Figure 1 – Number of detections

Рисунок 2 – Наиболее часто обнаруживаемые виды вредоносного ПО
Figure 2 – TOP of Malware

Как видно из «TOP угроз» на Рисунке 2, часто пользователь подвергается воздействию неизвестных раннее образцов вредоносного ПО. Таким образом, существующие антивирусные средства не гарантируют высокую степень защиты от новых, неизученных типов вредоносного ПО, так как требуют постоянной актуализации базы угроз [2]. Машинное обучение, а именно использование искусственных нейронных сетей, имеет большой потенциал в возможности обнаружения вредоносного кода, его обобщения и выявления связей с новыми семействами вредоносного ПО, а также позволит снизить риск ложных срабатываний на легальное программное обеспечение [3, 4]. Дополнительным обоснованием использования искусственных нейронных сетей для антивирусной защиты является то, что они могут обеспечить неявную классификацию двоичных файлов.

В связи с вышеизложенным целью исследования является разработка системы обнаружения вредоносного кода на основе машинного обучения.

Материалы и методы (Materials and Methods)

Очень важным этапом в построении антивирусной нейронной сети является ее обучение. Для решения этой задачи целесообразно использовать контролируемое обучение (обучение с учителем), так как существует большой массив данных: вредоносного и безопасного ПО. В основном, файлы, которые содержат вредоносный код, являются исполняемыми. Например, в операционной системе Windows это PE-файлы (.exe, .dll, .sys, .drv, .com, .ocx и т. д.) [4, 5].

В ходе исследования проведен анализ структуры исполняемых PE-файлов операционной системы Windows, выбран метод анализа кода, а также топология и характеристики нейронной сети.

Для задач классификации ПО на вредоносное и безопасное предлагается использовать нейронную сеть с прямой связью – четырёхуровневый персептрон [3]. Нейронная сеть с такой архитектурой продемонстрирована на Рисунке 3.

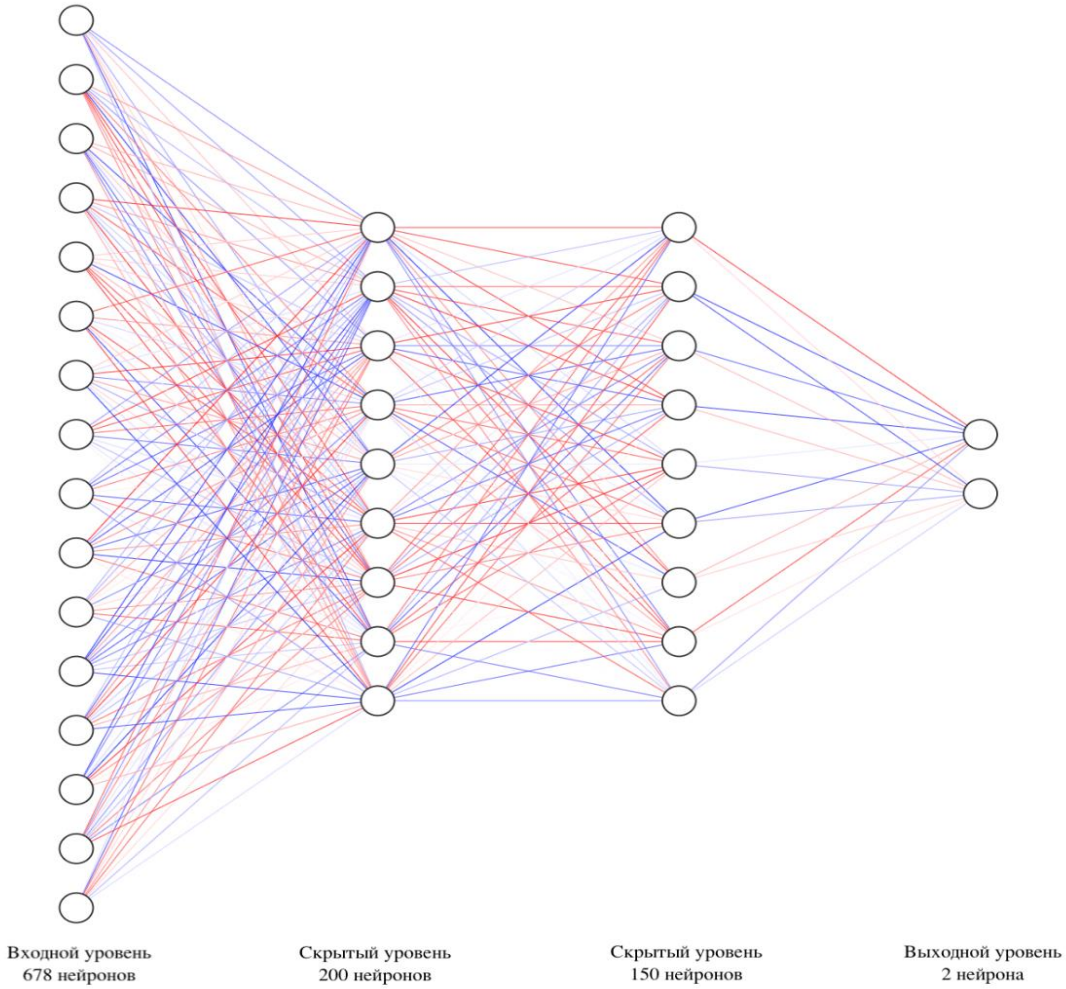

Рисунок 3 – Архитектура нейронной сети
Figure 3 – Neural network architecture

На входной уровень нейронной сети поступают нормализованные данные. В общем виде формула нормализации выглядит так:

$$y = \frac{(x - x_{min})(d2 - d1)}{x_{max} - x_{min}} + d1, \quad (1)$$

где x – значение подлежащие нормализации, $d1$ и $d2$ интервал, к которому будет приведено значение x.

На двух скрытых слоях используется функция активации PReLu, так как данная функция позволяет достичь высокой скорости обучения и не имеет недостатков классической ReLu. Для выходного уровня применяется сигмовидная функция активации. В качестве функции потерь применяется функция бинарной кросс-энтропии. Для выходного слоя нейронной сети при использовании функции потерь бинарной кросс-энтропии рекомендуется использовать сигмовидную функцию активации [6]. На практике такой подход позволил достичь более быстрой сходимости нейронной сети. Для оптимизации обучения нейронной сети применяется алгоритм Adam.

Для обучения нейронной сети из базы Ember [7] были получены характеристики 528 000 образцов безопасного и вредоносного ПО. Извлечённые данные были преобразованы в 528 000 векторов по 678 характеристик в каждом. В результате был получен тензор размерностью [528 000; 678].

Для извлечения характеристик из PE-файлов использовалась библиотека LIEF [8].

При формировании обучающего множества использовались следующие характеристики: байтовая гистограмма, гистограмма энтропии байтов [3], основная информация из PE-файла (наличие цифровой подписи, информации для отладки и т.д.), строки (наличие ссылок на веб-сайты и разделы реестра), информация об импортах и информация из заголовков секций PE-файла.

Для того, чтобы не было эффекта переобучения, тензор был разбит на 3 дополнительных тензора, предназначенных для обучения, проверки и тестирования. С помощью тензора проверки модель проверяется после обучения на тренировочных данных, в результате чего можно сделать вывод о соответствии точности модели по отношению к тренировочным данным, а также выполнить раннюю остановку обучения при увеличении ошибки прогнозирования. Финальная обученная модель нейронной сети проверяется на тензоре тестирования. Данный подход позволит узнать точность прогнозирования модели на раннее не известных модели образцах.

Однако всё равно есть большая вероятность возникновения эффекта переобучения, который появляется из-за перенасыщения нейронной сети однообразными обучающими выборками. В стандартной нейронной сети производная, полученная каждым параметром, сообщает блоку, как он должен измениться, чтобы минимизировать функцию конечных потерь с учетом деятельности остальных элементов. Поэтому блоки могут изменяться, исправляя при этом ошибки. Это может привести к чрезмерной совместной адаптации (co-adaptation), что, в свою очередь, приводит к переобучению. В качестве метода для решения данной проблемы можно обучать вместо одной сразу несколько нейронных сетей. Новые нейронные сети получаются на основе случайного исключения из сети нейронов. «Исключение» нейрона означает, что при любых входных данных или параметрах он возвращает 0. Для конечной нейронной сети следует использовать усреднённые полученные результаты. Метод, содержащий описанную идею, называется Dropout и хорошо показал на практике [9]. Для модели антивирусной нейронной сети применялся метод Dropout с 30% исключением в двух скрытых слоях.

Для реализации и обучения модели антивирусной нейронной сети выбран Keras (высокоуровневый интерфейс-надстройка для библиотеки TensorFlow). Код объявления модели для обучения на языке программирования Python в Keras:

```
Sequential([
Dense(200, input_shape=(n_inputs,)),
PReLU(),
Dropout(0.3),
Dense(150),
PReLU(),
Dropout(0.3),
Dense(2, activation='sigmoid')])
```

Код объявления модели для классификации на языке программирования Python в Keras:

```
Sequential([
Dense(200, input_shape=(n_inputs,)),
PReLU(),
Dense(150),
PReLU(),
Dense(2, activation='sigmoid')])
```

В Keras, как правило, не используется понятие входного слоя нейронов. Для этой цели в предложенной модели служит вектор значений (частный случай тензора),

значения которого предварительно нормализуются. Код объявленной модели соответствует Рисунку 3. Количество нейронов в скрытых слоях может быть рассчитано, исходя из формул (2) – (4).

$$r = \sqrt[3]{\frac{n}{m}}, \quad (2)$$

$$k1 = mr^2, \quad (3)$$

$$k2 = mr\ , \quad (4)$$

где $k1$ – число нейронов в первом скрытом слое, $k2$ – число нейронов во втором скрытом слое.

Существует еще более точный, но и более долгий способ. Он состоит в том, что сначала используется сеть с одним скрытым слоем с одним, двумя нейронами. Если сеть смогла достигнуть необходимого уровня ошибки, то процесс обучения закончен, иначе добавляем еще один нейрон и так до тех пор, пока ошибка сети не станет приемлемо малой, или до тех пор, пока увеличение числа нейронов не сможет значительно улучшить характеристики сети. Для конечной модели использовались представленные формулы и подход с экспериментом.

Для взаимодействия с библиотекой Keras, визуализации результатов обучения нейронной сети использовался программный инструмент Ergo [10]. После объявления модели, необходимо выполнить компиляцию модели. Код на языке программирования Python представлен ниже:

```
loss = 'binary_crossentropy'
optimizer = 'adam'
metrics = ['accuracy']
model.compile(loss = loss, optimizer = optimizer, metrics = metrics)
```

Результаты (Results)

Результаты обучения нейронной сети представлены на Рисунках 4-8.

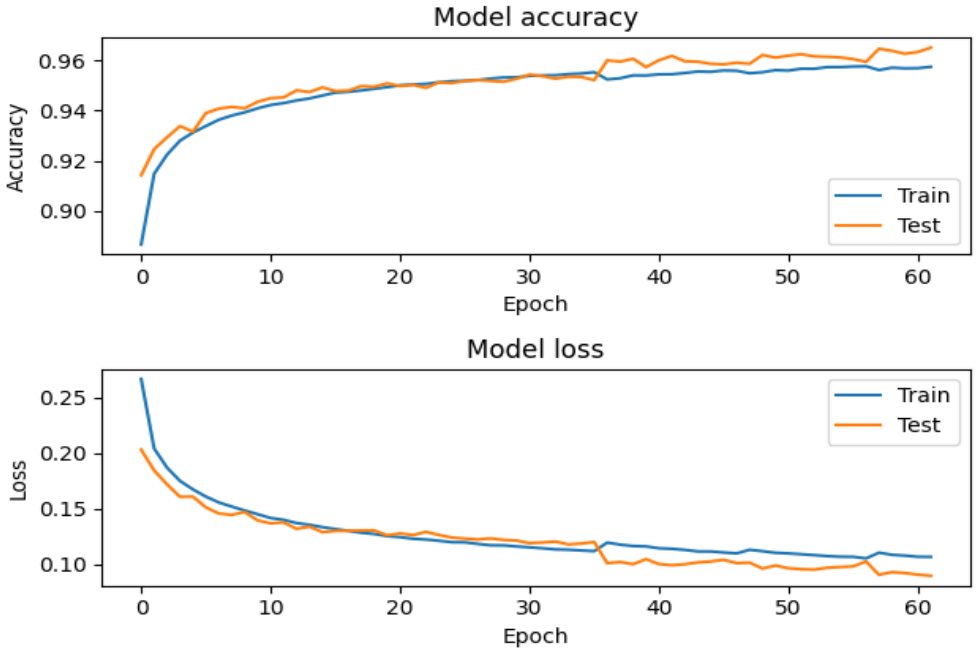

Рисунок 4 – История обучения нейронной сети
Figure 4 – Neural network training history

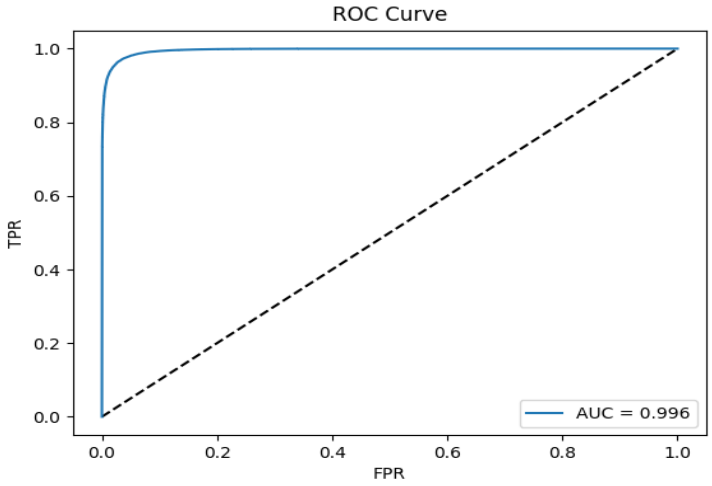

Рисунок 5 – ROC-кривая
Figure 5 – ROC Curve

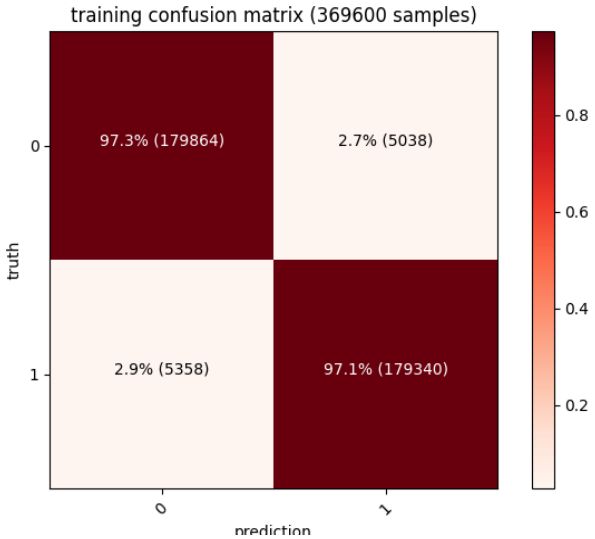

Рисунок 6 – Матрица ошибок на тренировочном наборе
Figure 6 – Training confusion matrix

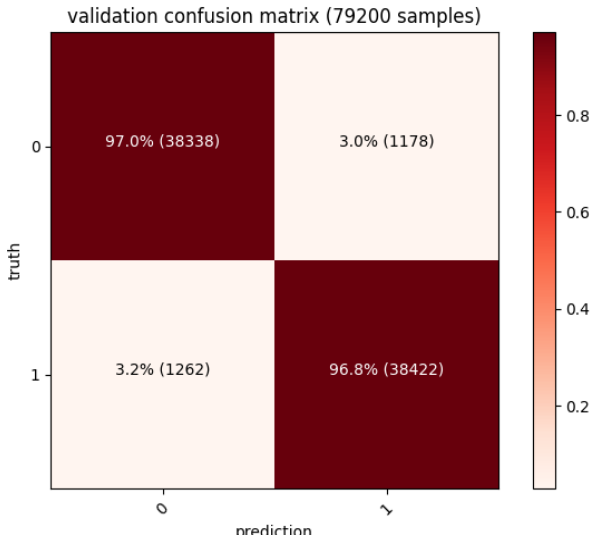

Рисунок 7 – Матрица ошибок на проверяющем наборе
Figure 7 – Validation confusion matrix

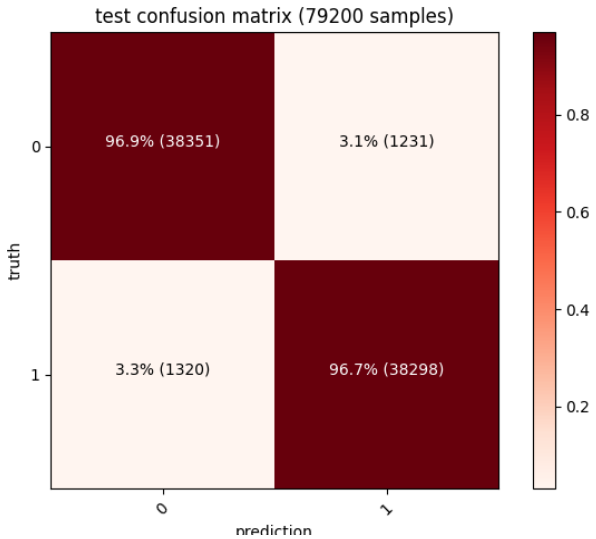

Рисунок 8 – Матрица ошибок на тестирующем наборе
Figure 8 – Test confusion matrix

Обсуждение (Discussion)

Графики истории обучения позволяют увидеть рост и падение показателей точности и ошибки нейронной сети. С ростом количества эпох обучения точность классификации стабильно росла. Факт более высокой точности классификации на тестовых данных демонстрирует отсутствие эффекта переобучения нейронной сети. ROC-кривая демонстрирует качество полученного бинарного классификатора, количественной характеристикой для данного графика является показатель AUC (площадь под ROC-кривой). Значение AUC, приближенное к 1, говорит об о высоком уровне классификации, выполняемом моделью. По диаграммам матриц ошибок можно судить о процентах образцов, которые были неверно распознаны моделью нейронной сети. В красных квадратах отображаются True Positive и True Negative результаты, в светлых False Positive и False Negative соответственно. Таким образом, на основании диаграммы тестирующего набора, можно сделать вывод, что модель всего в 3,1% случаев выдала False Positive результат на тестовые безопасные образцы ПО. Общее время обучения нейронной сети заняло 1 час, при этом процесс классификации ПО занимает миллисекунды.

Заключение (Conclusion)

В ходе исследования была выбрана архитектура нейронной сети – четырёхуровневый персептрон, были подобраны функции активации, функции потерь и алгоритм оптимизации сети, было рассчитано количество нейронов скрытых слоёв нейросети.

За время обучения нейронной сети достигнута высокая скорость сходимости при низкой ошибке в классификации PE-файлов. На тестовом наборе данных скомпилированная модель нейронной сети выявила около 97 % вредоносного ПО на тестовых данных, что подтверждается полученными диаграммами.

На данный момент, вследствие относительно небольшого количества данных для обучения, экспериментальная модель нейронной сети пока не способна полностью заменить антивирусные сканеры. Может потребоваться доработка и пересмотр процедуры извлечения характеристик из исполняемых файлов. Однако применение

антивирусной нейронной сети совместно со стандартными технологиями статического и эвристического анализов имеет большие перспективы.

ЛИТЕРАТУРА

1. Статистика. *Kaspersky Securelist*. Доступно по: https://statistics.securelist.com/ (Дата обращения: 30.08.2020).
2. Назаров А.В., Марьенков А.Н., Калиев А.Б. Выявление поведенческих признаков работы вируса-шифровальщика на основе анализа изменений значений параметров компьютерной системы. *Прикаспийский журнал: управление и высокие технологии*. 2018;1(41):196-204.
3. Saxe J., Berlin K. Deep Neural Network Based Malware Detection Using Two-Dimensional Binary Program Features. *Proceedings of 10th International Conference on Malicious and Unwanted Software (MALWARE)*. 2015. Доступно по: https://arxiv.org/pdf/1508.03096v2.pdf DOI: 10.1109/MALWARE.2015.7413680 (Дата обращения: 29.08.2020).
4. Пидченко И.А., Выборнова О.Н. Применение машинного обучения совместно с эвристическим анализом для задач антивирусного сканирования. *Математические методы в технике и технологиях – ММТТ*. 2020;5:96-99.
5. PE Format. Доступно по: https://docs.microsoft.com/en-us/windows/win32/debug/pe-format (Дата обращения: 29.08.2020).
6. Binary crossentropy. *Peltarion*. Доступно по: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/binary-crossentropy (Дата обращения: 30.08.2020).
7. Ember Dataset. Доступно по: https://github.com/endgameinc/ember (Дата обращения: 10.09.2020).
8. Library to Instrument Executable Formats. Доступно по: https://lief.quarkslab.com (Дата обращения: 10.09.2020).
9. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 2014;15:1929-1958. Доступно по: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf (Дата обращения: 29.08.2020)
10. Ergo. Доступно по: https://github.com/evilsocket/ergo (Дата обращения: 10.09.2020)

REFERENCES

1. Statistics. *Kaspersky Securelist*. Available from: https://statistics.securelist.com/ (Accessed 30th August 2020).
2. Nazarov A.V., Marenkov A.N., Kaliev A.B. Detection of cryptographic viruses behavior signs in the work of the computer system. *Caspian Journal: Management and high technologies*. 2018;1(41):196-204. (In Russ)
3. Saxe J., Berlin K. Deep Neural Network Based Malware Detection Using Two-Dimensional Binary Program Features. *Proceedings of 10th International Conference on Malicious and Unwanted Software (MALWARE)*. 2015. Available from: https://arxiv.org/pdf/1508.03096v2.pdf DOI: 10.1109/MALWARE.2015.7413680 (Accessed 29th August 2020).
4. Pidchenko I.A., Vybornova O.N. Application of machine learning in together with heuristic analysis for anti-virus scanning tasks. *Matematicheskie metody v tehnike i tehnologijah – MMTT*. 2020;5:96-99. (In Russ)

5. PE Format. Available from: https://docs.microsoft.com/en-us/windows/win32/debug/pe-format (Accessed 29th August 2020).
6. Binary crossentropy. *Peltarion*. Available from: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/binary-crossentropy (Accessed 30th August 2020).
7. Ember Dataset. Available from: https://github.com/endgameinc/ember (Accessed 10th September 2020).
8. Library to Instrument Executable Formats. Available from: https://lief.quarkslab.com (accessed 10th September 2020).
9. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 2014;15:1929-1958. Available from: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf (Accessed 29th August 2020)
10. Ergo. Available from: https://github.com/evilsocket/ergo (Accessed 10th September 2020)

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Выборнова Ольга Николаевна, доцент кафедры информационной безопасности, к.т.н., Астраханский государственный университет, Астрахань, Российская Федерация
email: *olga.vyb.90@gmail.com*
ORCID: 0000-0001-9458-1093

Olga N. Vybornova, Associate Professor of the Department of Information Security, candidate of technical sciences, Astrakhan State University, Astrakhan , Russian Federation

Пидченко Игорь Александрович, специалист по защите информации, ООО «Экономатика», Москва, Российская Федерация
email: *igor.pidchenko@gmail.com*

Igor A. Pidchenko, information security specialist, «Ekonomatika OOO» (limited liability company), Moscow, Russian Federation