Software Impacts 13 (2022) 100365

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

PE Parser: A Python package for Portable Executable files processing

Daniel Gibert

CeADAR, University College Dublin, Ireland

ARTICLE INFO

Keywords:
Malware classification
Feature engineering
Grayscale representation
Structural entropy
PE executables

ABSTRACT

PE Parser is a Python package to parse and work with the hexadecimal representation of executables' binary content and its assembly language source code. *PE Parser* has been designed to provide a class-based and user-friendly interface for the extraction of well-known features commonly used for the task of malware detection and classification such as byte and opcode N-Grams, API function calls, the frequency of use of the registers, characteristics of the Portable Executable file sections, among others. In addition, *PE Parser* has various command line tools to visualize the executables as grayscale images or as a stream of entropy values.

Code metadata

Current code version	v1.0.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2022-131
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/9987564/tree/v1
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	Python, NLTK
Compilation requirements, operating environments & dependencies	Python>= 3.6.9
If available Link to developer documentation/manual	https://pe-parser.readthedocs.io/en/latest/
Support email for questions	daniel.gibertlla@gmail.com

1. Introduction

Feature engineering is a key component of any machine learning system's pipeline. Broadly speaking, feature engineering refers to the process of transforming raw data into meaningful features used to feed the desired learning algorithm. For the task of malware detection and classification, it is common to extract features from both the binary content of a Portable Executable (PE) file and its corresponding assembly language source code. Common features are byte and opcode N-gram features, entropy statistics, the frequency of use of the registers, the invocation or not of Application Programming Interface (API) functions and system calls, etcetera. Afterwards, the resulting features are used to train a machine learning model [1–4].

PE Parser is a Python package for the preprocessing, the extraction of features and the visualization of Portable Executable files, both as grayscale images [5,6] and as a stream of entropy values [7,8]. Existing Python libraries, such as EMBER [9] and PE Miner [10], extract features and characteristics from the Portable Executable files headers, i.e MS-DOS Header, COFF file header, and the Optional header. Additionally, *PE Parser* complements the aforementioned libraries by providing various procedures to extract well-known features from the assembly language source code of PE files. *PE Parser* has been designed to provide a class-based and user-friendly interface to facilitate the extraction of well-known features [1,2] from Portable Executables files that are commonly used to build malware detection systems powered by machine learning. Furthermore, *PE Parser* provides various command line tools to visualize the executables as grayscale images [5,6,11] or as a stream of entropy values (structural entropy) [7,8].

2. Key features

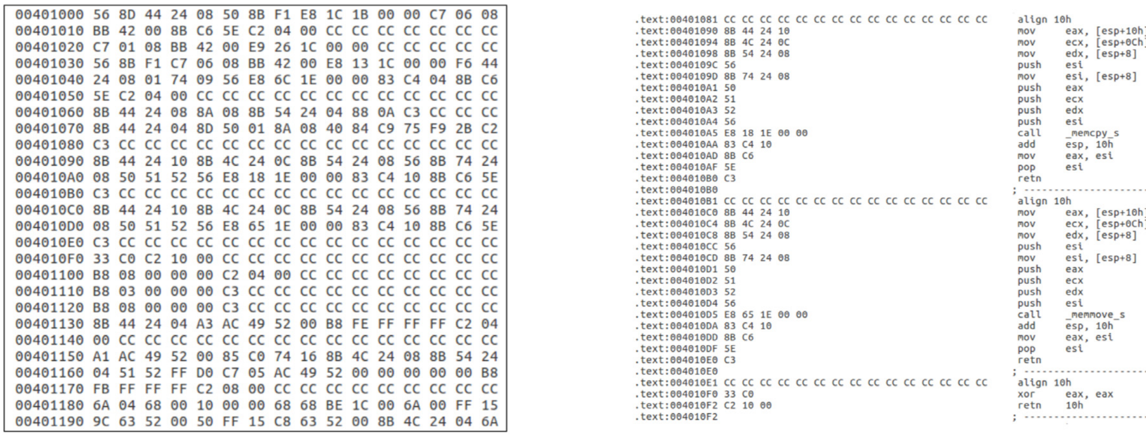
PE Parser can be used to extract features from Portable Executable files in one of the following two formats:

- Hexadecimal representation of its binary content. Cf. Fig. 1(a). This representation represents the machine code as a sequence of hexadecimal values. The first value indicates the starting address

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

E-mail address: daniel.gibert@ucd.ie.

https://doi.org/10.1016/j.simpa.2022.100365
Received 30 June 2022; Received in revised form 8 July 2022; Accepted 11 July 2022



(a) Hexadecimal view.

(b) Assembly view.

Fig. 1. Hexadecimal and assembly view of PE executables [13].

of the machine codes in the memory, and each hexadecimal value (byte) carries meaningful information of the Portable Executable file such as instruction codes and data. There are various tools to obtain the hexadecimal view of a binary file such as PE Explorer,[1] HxD,[2] among others.

- Assembly language source code. Cf. Fig. 1(b). The assembly language source code contains the symbolic machine code of the executable as well as metadata information such as rudimentary function calls, memory allocation and variable information. There are various tools for disassembling Portable Executable files such as IDA Pro,[3] Radare2,[4] Ghidra,[5] etcetera.

These file formats are the ones commonly used during static analysis to extract features from executables without actually running the program. For more information about both file formats we refer the readers to the Microsoft Malware Classification Challenge dataset [12], a high-quality public labeled benchmark available for malware classification research. This dataset has become the standard benchmark to evaluate machine learning approaches for malware classification and currently, it is publicly available in the Kaggle platform.[6]

Currently, *PE Parser* contains code to extract the following features:

- Hexadecimal-based features. Features extracted from the hexadecimal representation of executable's binary content.
  - Byte N-gram features.
  - Entropy-based features.
  - Haralick features from the grayscale image representation of the hexadecimal view of executables.
  - Local Binary Pattern features from the grayscale image representation of the hexadecimal view of executables.
- Assembly-based features.
  - Opcode N-gram features.
  - Register features.
  - Data define directive features.
  - Section characteristics features.
  - Symbol frequency features.
  - Application Programming Interface (API) function calls features.
  - Pixel intensity features extracted from the grayscale representation of the assembly code.
  - Miscellaneous features (Keywords).

A complete description of the aforementioned features is provided in the works of Ahmadi et al. [1], Zhang et al. [2], and Gibert et al. [4].

3. Visualization tools

PE Parser provides various command-line tools to visualize the executables as follows:

Grayscale image representation. Nataraj et al. [5] proposed to visualize and classify malware (in PE format) using image processing techniques. Cf. Fig. 2. In their work, binary executables are visualized as grayscale images, where every byte is reinterpreted as one pixel in the image. Then, the resulting array is reorganized as a 2-D array and can be visualized as an image, with values ranging from 0 to 255.

This grayscale image representation can be used to classify malware as images of malware belonging to the same family are similar between them while distinct from those belonging to other families as demonstrated in the work of Gibert et al. [6] and others [14].

Structural entropy representation. Portable Executable files can be visualized as a stream of entropy values, or structural entropy [7,8]. To calculate the structural entropy representation of an executable, you need to split the executable into chunks of fixed size, i.e. 1024 bytes, and calculate the amount of entropy for each chunk. Cf. Fig. 3.

4. Impact

PE Parser is intended for scientists and researchers who would like to conduct research on malware detection. *PE Parser* has been designed to be used by both computer security researchers and by students in courses on malware analysis, machine learning and computer security. Currently, it has been already used in a number of scientific publications [3,6,8,13,15–17]. Besides, the authors are aware of numerous security practitioners that have used one or more of the features implemented by PE Parser to build their malware detection systems [1,2]. In

[1] http://www.heaventools.com/flexhex-hex-editor.htm
[2] https://mh-nexus.de/en/hxd/
[3] https://www.hex-rays.com/products/ida/
[4] https://rada.re/n/
[5] https://ghidra-sre.org/
[6] https://www.kaggle.com/c/malware-classification

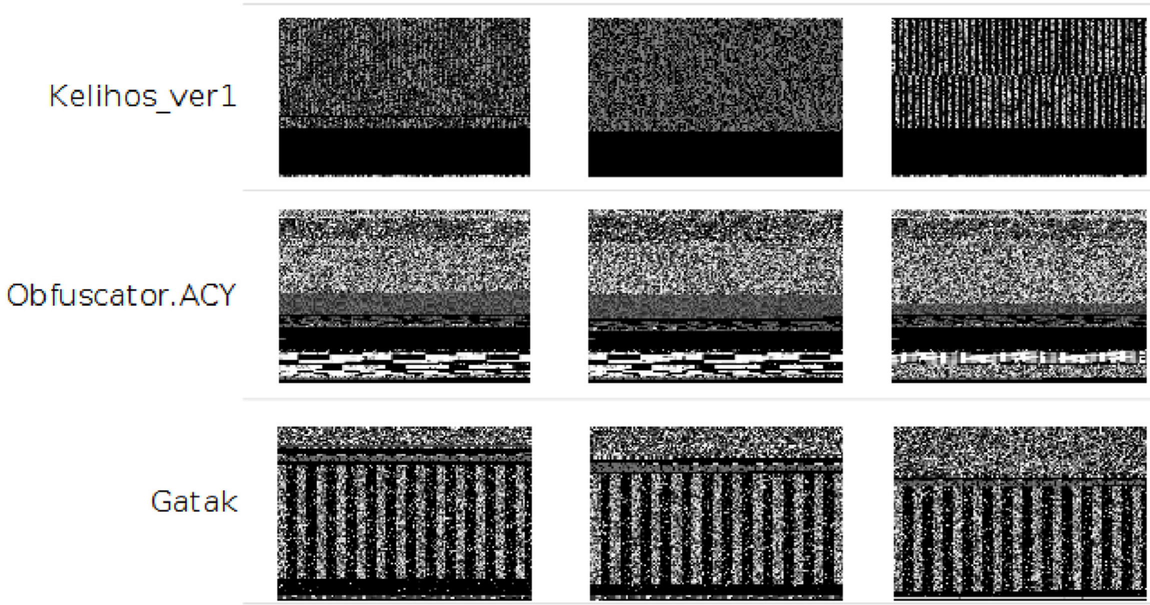


Fig. 2. Grayscale image representation of samples belonging to the Kelihos_ver1, Obfuscator.ACY and Gatak families [6].

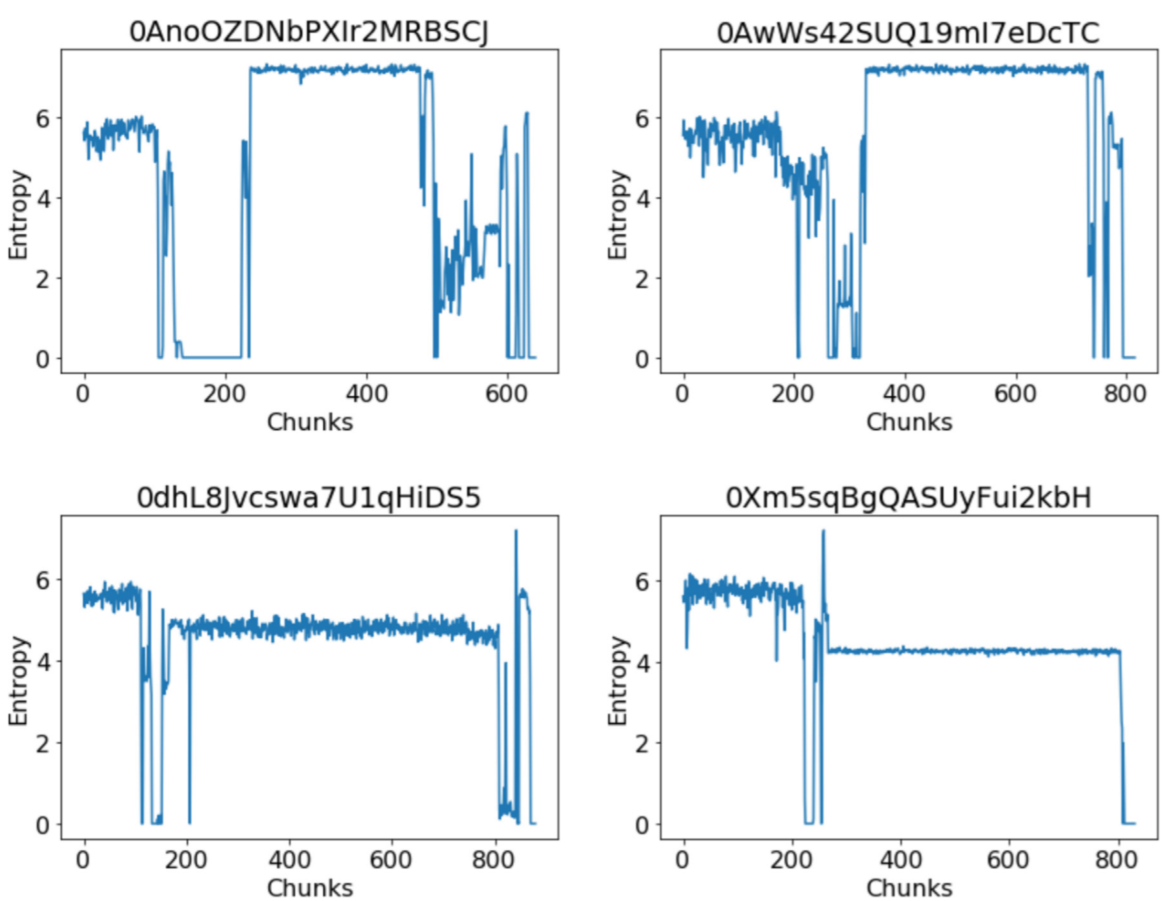


Fig. 3. Structural entropy representation of samples belonging to the Ramnit and Gatak families [8].

addition, the representation of malware's binary content as a grayscale image has been used in at least the following papers [5,6,18–27] while the structural entropy representation of malware has been used in at least next papers [7,8,28–36]. Furthermore, *PE Parser* can be used to train deep learning models [3,15,37–41] by using as input the hexadecimal byte values and the opcodes extracted from the hexadecimal representation of malware's binary content and its assembly language source code, respectively.

5. Future work

In the future we intend to extend the library with additional types of features and visualization tools such as entropy graphs [36] and function call graphs [42]. On the one hand, entropy graphs can be created using a two steps process. First, binary files are converted to bitmap images. Second, the entropy value of each line of the bitmap images is calculated. On the other hand, the function call graph is a control-flow graph that represents calling relationships between functions/subroutines in a computer program.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 847402.

References

[1] Mansour Ahmadi, Dmitry Ulyanov, Stanislav Semenov, Mikhail Trofimov, Giorgio Giacinto, Novel feature extraction, selection and fusion for effective malware family classification, in: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, in: CODASPY, vol. 16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 183–194.

[2] Y. Zhang, Q. Huang, X. Ma, Z. Yang, J. Jiang, Using multi-features and ensemble learning method for imbalanced malware classification, in: 2016 IEEE Trustcom/BigDataSE/ISPA, 2016, pp. 965–973.

[3] Daniel Gibert, Carles Mateu, Jordi Planes, HYDRA: A multimodal deep learning framework for malware classification, Comput. Secur. 95 (2020) 101873.

[4] Daniel Gibert, Jordi Planes, Carles Mateu, Quan Le, Fusing feature engineering and deep learning: A case study for malware classification, Expert Syst. Appl. (2022).

[5] L. Nataraj, S. Karthikeyan, G. Jacob, B.S. Manjunath, Malware images: Visualization and automatic classification, in: Proceedings of the 8th International Symposium on Visualization for Cyber Security, in: VizSec, vol. 11, ACM, New York, NY, USA, 2011, pp. 4:1–4:7.

[6] Daniel Gibert, Carles Mateu, Jordi Planes, Ramon Vicens, Using convolutional neural networks for classification of malware represented as images, J. Comput. Virol. Hack. Tech. (2018).

[7] Donabelle Baysa, Richard M. Low, Mark Stamp, Structural entropy and metamorphic malware, J. Comput. Virol. Hacking Tech. 9 (4) (2013) 179–192.

[8] Daniel Gibert, Carles Mateu, Jordi Planes, Ramon Vicens, Classification of malware by using structural entropy on convolutional neural networks, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI-18, New Orleans, Louisiana, USA, February 2-7, 2018, 2018, pp. 7759–7764.

[9] Hyrum S. Anderson, Phil Roth, EMBER: An open dataset for training static PE malware machine learning models, 2018, CoRR, abs/1804.04637.

[10] M. Zubair Shafiq, S. Momina Tabish, Fauzan Mirza, Muddassar Farooq, PE-Miner: Mining structural information to detect malicious executables in realtime, in: Engin Kirda, Somesh Jha, Davide Balzarotti (Eds.), Recent Advances in Intrusion Detection, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 121–141.

[11] B.N. Narayanan, O. Djaneye-Boundjou, T.M. Kebede, Performance analysis of machine learning and pattern recognition algorithms for malware classification, in: 2016 IEEE National Aerospace and Electronics Conference, NAECON and Ohio Innovation Summit (OIS), 2016, pp. 338–342.

[12] Royi Ronen, Marian Radu, Corina Feuerstein, Elad Yom-Tov, Mansour Ahmadi, Microsoft malware classification challenge, 2018, CoRR abs/1802.10135.

[13] D. Gibert, C. Mateu, J. Planes, Orthrus: A bimodal learning architecture for malware classification, in: 2020 International Joint Conference on Neural Networks, IJCNN, 2020, pp. 1–8.

[14] Xinbo Liu, Yaping Lin, He Li, Jiliang Zhang, A novel method for malware detection on ML-based visualization technique, Comput. Secur. 89 (2020) 101682.

[15] D. Gibert, J. Bejar, C. Mateu, J. Planes, D. Solis, R. Vicens, Convolutional neural networks for classification of malware assembly code, in: International Conference of the Catalan Association for Artificial Intelligence, 2017, pp. 221–226.

[16] Daniel Gibert, Carles Mateu, Jordi Planes, The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, J. Netw. Comput. Appl. 153 (2020) 102526.

[17] Daniel Gibert, Carles Mateu, Jordi Planes, Joao Marques-Silva, Auditing static machine learning anti-malware tools against metamorphic attacks, Comput. Secur. 102 (2021) 102159.

[18] Yusheng Dai, Hui Li, Yekui Qian, Xidong Lu, A malware classification method based on memory dump grayscale image, Digit. Investig. 27 (2018) 30–37.

[19] Mazhar Javed Awan, Osama Ahmed Masood, Mazin Abed Mohammed, Awais Yasin, Azlan Mohd Zain, Robertas Damaševičius, Karrar Hameed Abdulkareem, Image-based malware classification using VGG19 network and spatial convolutional attention, Electronics 10 (19) (2021).

[20] Aziz Makandar, Anita Patrot, Malware class recognition using image processing techniques, in: 2017 International Conference on Data Management, Analytics and Innovation, ICDMAI, 2017, pp. 76–80.

[21] Ke He, Dong-Seong Kim, Malware detection with malware images using deep learning techniques, in: 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE, 2019, pp. 95–102.

[22] Danish Vasan, Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei, Qin Zheng, IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture, Comput. Netw. 171 (2020) 107138.

[23] Yifei Jian, Hongbo Kuang, Chenglong Ren, Zicheng Ma, Haizhou Wang, A novel framework for image-based malware detection with a deep neural network, Comput. Secur. 109 (2021) 102400.

[24] Wai Weng Lo, Xu Yang, Yapeng Wang, An xception convolutional neural network for malware classification with transfer learning, in: 2019 10th IFIP International Conference on New Technologies, Mobility and Security, NTMS, 2019, pp. 1–5.

[25] Danish Vasan, Mamoun Alazab, Sobia Wassan, Babak Safaei, Qin Zheng, Image-based malware classification using ensemble of CNN architectures (IMCEC), Comput. Secur. 92 (2020) 101748.

[26] Anson Pinhero, Anupama M L, Vinod P, C.A. Visaggio, Aneesh N, Abhijith S, AnanthaKrishnan S, Malware detection employed by visualization and deep neural network, Comput. Secur. 105 (2021) 102247.

[27] Sitalakshmi Venkatraman, Mamoun Alazab, R. Vinayakumar, A hybrid deep learning image-based analysis for effective malware detection, J. Inf. Secur. Appl. 47 (2019) 377–389.

[28] Gerardo Canfora, Francesco Mercaldo, Corrado Aaron Visaggio, An HMM and structural entropy based detector for android malware: An empirical study, Comput. Secur. 61 (2016) 1–18.

[29] Alfredo Cuzzocrea, Fabio Martinelli, Francesco Mercaldo, A novel structural-entropy-based classification technique for supporting android ransomware detection and analysis, in: 2018 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE, 2018, pp. 1–7.

[30] Guoqing Xiao, Jingning Li, Yuedan Chen, Kenli Li, MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks, J. Parallel Distrib. Comput. 141 (2020) 49–58.

[31] Hui Guo, Shuguang Huang, Cheng Huang, Zulie Pan, Min Zhang, Fan Shi, File entropy signal analysis combined with wavelet decomposition for malware classification, IEEE Access 8 (2020) 158961–158971.

[32] Joon-Young Paik, Rize Jin, Eun-Sun Cho, Malware classification using a byte-granularity feature based on structural entropy, Comput. Intell. n/a (n/a) (2022).

[33] Michael Wojnowicz, Glenn Chisholm, Matt Wolff, Xuan Zhao, Wavelet decomposition of software entropy reveals symptoms of malicious code, J. Innov. Digit. Ecosyst. 3 (2) (2016) 130–140.

[34] Munkhbayar Bar-Erdene, Hyundo Park, Hongzhe Li, Heejo Lee, Mahn-Soo Choi, Entropy analysis to classify unknown packing algorithms for malware detection, Int. J. Inf. Secur. 16 (2017).

[35] Esmaeel Radkani, Sattar Hashemi, Alireza Keshavarz-Haddad, Maryam Amir Haeri, An entropy-based distance measure for analyzing and detecting metamorphic malware, Appl. Intell. 48 (2018).

[36] Kyoung Soo Han, Jae Hyun Lim, Boojong Kang, Eul Gyu Im, Malware analysis using visualized images and entropy graphs, Int. J. Inf. Secur. 14 (2015).

[37] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, Charles K. Nicholas, Malware detection by eating a whole EXE, in: The Workshops of the the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018, in: AAAI Technical Report, vol. WS-18, AAAI Press, 2018, pp. 268–276.

[38] Marek Krcál, Ondrej Svec, Martin Bálek, Otakar Jasek, Deep convolutional malware classifiers can learn from raw executables and labels only, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings, OpenReview.net, 2018.

[39] Edward Raff, William Fleshman, Richard Zak, Hyrum S. Anderson, Bobby Filar, Mark McLean, Classifying sequences of extreme length with constant memory applied to malware detection, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, the Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 9386–9394.

[40] Li Yang, Junlin Liu, TuningMalconv: Malware detection with not just raw bytes, IEEE Access 8 (2020) 140915–140922.

[41] Quan Le, Oisín Boydell, Brian Mac Namee, Mark Scanlon, Deep learning at the shallow end: Malware classification for non-domain experts, Digit. Investig. 26 (2018) S118–S126.

[42] B.G. Ryder, Constructing the call graph of a program, IEEE Trans. Softw. Eng. SE-5 (3) (1979) 216–226.