MINING PATTERNS OF SEQUENTIAL MALICIOUS APIS TO DETECT MALWARE

Abdurrahman Pektaş[1], Elif Nurdan Pektaş[2] and Tankut Acarman[1]

[1]Department of Computer Engineering, Galatasaray University, İstanbul, Turkey
[2]Siemens Turkey, Yakack Caddesi No: 111, 34870 Kartal, Istanbul, Turkey

ABSTRACT

In the era of information technology and connected world, detecting malware has been a major security concern for individuals, companies and even for states. The New generation of malware samples upgraded with advanced protection mechanism such as packing, and obfuscation frustrate anti-virus solutions. API call analysis is used to identify suspicious malicious behavior thanks to its description capability of a software functionality. In this paper, we propose an effective and efficient malware detection method that uses sequential pattern mining algorithm to discover representative and discriminative API call patterns. Then, we apply three machine learning algorithms to classify malware samples. Based on the experimental results, the proposed method assures favorable results with 0.999 F-measure on a dataset including 8152 malware samples belonging to 16 families and 523 benign samples.

KEYWORDS

Android, Malware, Frequent Sequence Mining, Behavioural Pattern, API Calls, Dynamic Analysis

1. INTRODUCTION

A malware is deployed to execute malicious behaviours on the compromised system without authorization and knowledge of its user. A malware steals sensitive information, damages the integrity of the compromised system, and joins the compromised computer into a part of a cyber-crime as a boot and so on. A large variety of malware samples exists while targeting different information systems such as mobile phones (especially Android and IOS), IoT, MAC OS, Linux, etc. In this study, we focus on Windows based-malware since Microsoft OS is the far more the most popular desktop OS with a market share of about 82.86% [1].

Concerning security threats and trends, according to 2016 Symantec's Internet security report [2], 18.4 million mobile malware variants were detected. 4 new mobile families and clusters of 61 distinct new mobile threats were discovered in 2016 versus the identification of 18 new families and 75 clusters in 2015. The high number of malware samples versus change in growth rate and new family number can be interpreted such that malware writers have been focused on refining and modifying existing malware families in order to mitigate detection and execute malicious intents instead of deploying new and unique malware threats.

There are mainly two reasons about this volume. Firstly, the malware authors can easily access source code of malware in the Internet and build new cyber weapons. Secondly, malware writers make use of the runtime packer and obfuscation techniques, which easily enable them creation of behaviourally identical but statically different malware samples (called malware variants) [3]. To detect a malware, features are extracted by either static or dynamic analysis. Static features such as Application Programming Interface (API) calls, opcode sequences, permission requests, control flows or data flows are extracted from the static source code [4]. These features heavily depend on data distribution and a large dataset enables a comprehensive learning while introducing more promising detection performance. Dynamic features are extracted by execution of a given application and activities related to network, file system access and interaction with the system are monitored [5].

API calls invoked during execution of a program present malicious behaviours and functionality. Sequential pattern mining and frequent sequence mining is the technique used to discover meaningful knowledge in the sequential dataset. Sequential pattern mining has been applied successfully in a variety of data analysis requirements such as bioinformatics, text analysis and user modelling through mining web usage [6, 7]. In [8], frequent API names and their arguments are used to represent the behaviour of a malware. To capture the API calls, a malware is run in a sandbox environment. After getting API calls, frequent item set mining is applied. At the final stage, the malware dataset is classified according to their families. Based on the experiment result, 3131 malware samples belonging to 24 malware families are classified with 94.7% F-measure. But this study does not take into account the sequential nature of the API calls and particular focus has been on the frequent API calls.

In [9] to capture the common API sequence among different malware categories, the longest common sequence (LCS) algorithm is used. The captured sequences are treated as the signature. Similar to the previous work, [10] uses multiple sequence alignment (MSA) algorithms to extract representative API call patterns of malware families. API call sequences are extracted through a dynamic analysis approach. After applying MSA to the API calls, the similarity calculation is performed to classify malware into their respective families.

The main limitation of these methods is that extraction of the longest common API sequence is a an NP-hard problem [11, 12]. In other words, the time complexity of these methods is exponentially increasing versus the size of the dataset. [13] introduces a malware detection approach based on the malicious instruction patterns. The work includes three major steps: 1) extraction of instruction sequence via disassembling samples, 2) mining malicious instruction patterns using Generalized Sequential Pattern (GSP) algorithm, and 3) classification of malware families using the All-Nearest- Neighbour algorithm. A recent study [14] employs the deep neural network to acquire the representative and distinguishing API call patterns of malware families. Again, the dynamic analysis method is used to collect API calls. LSTM neural network is applied to the collected API calls from 787 malware samples belonging to 9 families. The classification accuracy is reached at 71% on average.

In [15], Canfora et al. presents a mobile malware detection approach based on Hidden Markov Model and structural entropy. The proposed approach was evaluated on a balanced dataset including 5560 malware and 5560 trusted Android applications. The experimental results reached at 98% precision. In [16], Salehi et al employ dynamic feature collected by running malicious executable into controlled environment. Essentially, the authors use API calls, their arguments and return values. After applying feature selection on the entire feature set, selected features achieved 98.1% f-measure on the dataset which consists of 3009 malicious and 1359 benign PE files. [17] integrates static and dynamic features to identify malware samples. SVM algorithm achieved more accurate results than Random Forest with 98.7% accuracy. The authors also show that the combination of static and dynamic features provides the more accurate classification results.

In this paper we address the challenge of identifying the representative sequence of API calls that is used by malware authors. We mine API call sequence using sequential pattern mining approaches and extract meaningful and distinguishing API sub-sequence. Then we use these API calls to model a malware and apply machine learning algorithms to classify malware samples into their families. According to the experimental results conducted on a fairly large scale and publicly available dataset including 8,675 samples, the proposed method provides a promising solution to automatic classification of malware samples.

The remaining of the paper is organized as follows: the proposed method to detect malicious software by mining frequent sequential APIs is presented in Section 2. Section 3 introduces the dataset, experimental study and evaluation result. Finally, some conclusions are presented in Section 4.

2. METHODOLOGY

An overview of the system including its main functionalities is presented in Figure-1. The first stage is dynamic analysis, more precisely the behaviour of the sample application under analysis is observed and its reports its interaction with system resources. At this stage, the sample application is run in a sandboxed environment; Cuckoo [18] to monitor and collect behaviour of a given executable file. At the second stage, feature to the dynamic dynamic analysis reports. Then, each malware is modelled by employing the discovered sequence of API calls and each sample is labeled by using Virustotal [19]. At the final stage, three machine learning algorithms are evaluated by using a 10-fold cross-validation approach [20].

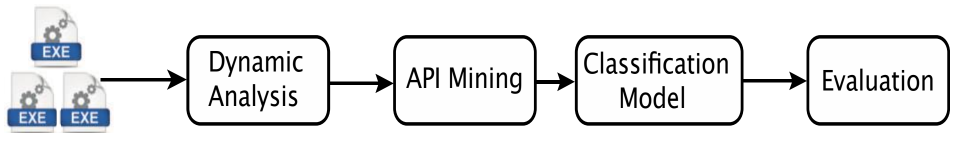


Figure 1. Overview of the proposed methodology

2.1. DYNAMIC ANALYSIS

API calls must be executed by malware authors in order to perform malicious activities and intents. In consequence, the behaviour of a sample file can be identified by malware analyst. In general terms, the behaviour of an executable file can be derived from the sequence of Windows API calls. System, registry, services and network activities constitute the main functional categories of the Windows API function calls. During dynamic analysis API calls are encoded into two length long codes (i.e., kind of words). Successive API calls are ignored during the encoding procedure. And the semantics of the API calls, which are resilient to obfuscation techniques, are captured by the encoding method in an effective manner.

2.2. MINING API CALLS

In Windows API, there are different versions for the same API call [21]. For example, ShellExecute and ShellExecuteEx are two different API functions. Once Windows upgrade an API, Ex suffix is added to the legacy function names and Microsoft continues to support the old APIs. Moreover, Microsoft generally uses two distinct API calls for ANSI and Unicode strings. A suffix is used for ANSI text as input and output, and the W suffix is used Unicode text. For example, SetWindowTextA and SetWindowTextW are supported for ANSI and Unicode strings, respectively. To normalize API calls suffix from API calls are removed.

After getting API call sequence from the dataset, we explore the frequent sequence of API calls to build feature set and then model our dataset. Essentially, sequential API mining reveals distinctive and representative subsequence in the set of API call sequence. Frequent sequences can be discovered by different approaches that rely on different measures such as length, frequency, and support. Generally, the support measure is preferred. The support of a sequence s_i is formally defined as the number of sequence in which s_i appears and denoted by $\sup(s_i)$. Sometimes the notation *relative support* is used. Relative support is defined as the ratio of the number of sequence containing s_i to the total number of sequences. Sequential pattern mining aims to find all frequent subsequences in the dataset [22, 23]. And the sequence s_i is frequent sequence if and only if support of s_i is greater than the threshold minimum support value(min_{sup}) given by the user, $sup(s_i) > min_{sup}$.

Discovering frequent sequences in a large-scale dataset is a challenging problem [24]. Various algorithms such as GSP [25] Spam [26] and Spade [27] have been proposed to efficiently explore the subsequence. One thing to note that, these algorithms take a sequence dataset and a minimum support (threshold value) and generate the frequent sequences. The frequent sequence discovery algorithms generate the same set of subsequence's. However, these algorithms differ in searching and navigating the sequential patterns and in utilizing the different data structures.

In our study, we use the SPMF data mining library, an open-source Java library includes more than 138 implementations of pattern discovery algorithms [28] (SPMF library is available at [29]). According to the experiment in [28], CM-SPAM algorithm is evaluated as the most successful sequential pattern mining algorithm in SPMF. Therefore, we choose CM-SPAM algorithm to discover sequential API patterns. Interested reader can refer to [30] for the CM-SPAM algorithm, which is evaluated to be faster than SPAM. We tune the following optional parameters in CM-SPAM implementation:

- **Relative support**: The ratio of the number of sequence containing s_i to the total number of sequences. This parameter is set to 50%.
- **Minimum pattern length**: This parameter specifies the minimum length of the API sequence. It is set to 4.
- **Maximum pattern length**: This parameter specifies the maximum length of the API sequence. It is set to 8.
- **Max gap**: This parameter is used to specify if gaps are allowed in sequential patterns. We set it to 1, which means no gap between each API calls.

2.3.BUILDING CLASSIFICATION MODEL

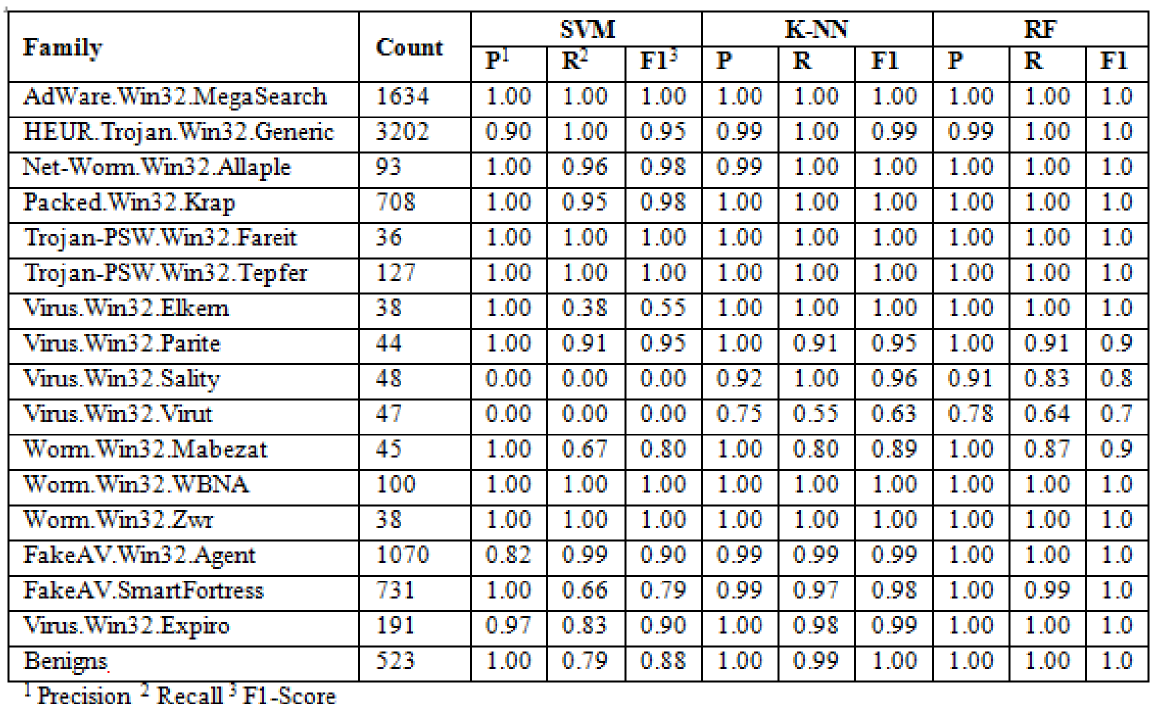
A supervised machine learning approach is adopted to classify malware samples like [31]. In the study, malicious API patterns generated in the previous step are considered as a feature to model and represent malware samples. We assign a weight to each API pattern by multiplying the inverse document frequency (idf) and term frequency (tf). The API features are then transformed into vector representation and each sample is characterized by its feature vector. Namely, each API call sequence is represented as the vector along with the product of idf and tf. Three well-known machine classification algorithms including Random Forest [32], K-NN [33] and SVM [34] are evaluated.

3. EVALUATION

We download the latest malware dataset from Virusshare and select the Windows executable files from the dataset. The malware dataset only includes exe files not dll or Windows installation file(msi). To identify and verify the Windows exe files, we utilize file command in linux. File command is aim to determine type of a given file by analyzing its content especially file's magic numbers. Magic number also known as file signature is the very first bunch of bytes in a file, which can uniquely specify the type of a file. Overall, we select 8152 number of malware samples. For the benign dataset, we utilize the exe files located under the C:/ directory of the fresh installation of the Windows 10 OS. The distribution of the dataset is given in Table-1, in which P denotes Precision, R denotes Recall and F1 denotes F1-measure statistical metric values, respectively.

To assess the effectiveness of the selected classifiers, we used four metrics: precision, recall (a.k.a. sensitivity), F1-score, overall accuracy. We conduct experiments using 10-fold cross-validation method. In this way, we prevent the over-fitting problem. All experiments are run on a regular PC environment with a 2-Core i-7 processor and 8 GB memory, using Scikit learn machine learning library [35]. Table-1 shows the recall, precision and F1-score of each machine learning algorithm when performing malware family classification. According to the numerical results related to each metric, Random Forest outperforms the other two classifiers and achieves the highest precision, recall and F-score for each family. The RF classifiers almost perfectly classify all families except Virut and Sality families. The main reason behind this is that these two malware families are very close to each other in terms of executing actions. Besides that, these two malware families belong to virus malware fam ily.

Table 1. Classification accuracies of the machine learning algorithms per malware families.



Family	Count	SVM			K-NN			RF		
		P[1]	R[2]	F1[3]	P	R	F1	P	R	F1
AdWare.Win32.MegaSearch	1634	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.0
HEUR.Trojan.Win32.Generic	3202	0.90	1.00	0.95	0.99	1.00	0.99	0.99	1.00	1.0
Net-Worm.Win32.Allaple	93	1.00	0.96	0.98	0.99	1.00	1.00	1.00	1.00	1.0
Packed.Win32.Krap	708	1.00	0.95	0.98	1.00	1.00	1.00	1.00	1.00	1.0
Trojan-PSW.Win32.Fareit	36	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.0
Trojan-PSW.Win32.Tepfer	127	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.0
Virus.Win32.Elkern	38	1.00	0.38	0.55	1.00	1.00	1.00	1.00	1.00	1.0
Virus.Win32.Parite	44	1.00	0.91	0.95	1.00	0.91	0.95	1.00	0.91	0.9
Virus.Win32.Sality	48	0.00	0.00	0.00	0.92	1.00	0.96	0.91	0.83	0.8
Virus.Win32.Virut	47	0.00	0.00	0.00	0.75	0.55	0.63	0.78	0.64	0.7
Worm.Win32.Mabezat	45	1.00	0.67	0.80	1.00	0.80	0.89	1.00	0.87	0.9
Worm.Win32.WBNA	100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.0
Worm.Win32.Zwr	38	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.0
FakeAV.Win32.Agent	1070	0.82	0.99	0.90	0.99	0.99	0.99	1.00	1.00	1.0
FakeAV.SmartFortress	731	1.00	0.66	0.79	0.99	0.97	0.98	1.00	0.99	1.0
Virus.Win32.Expiro	191	0.97	0.83	0.90	1.00	0.98	0.99	1.00	1.00	1.0
Benigns	523	1.00	0.79	0.88	1.00	0.99	1.00	1.00	1.00	1.0

[1] Precision [2] Recall [3] F1-Score

We also evaluate the effectiveness of each algorithm using the area under the receiver operating characteristic (ROC) curve, i.e. AUC. ROC curve is a plot of the TP rate against the FP rate for different decision thresholds [36]. The area under the ROC curve (AUC) measures the prediction capability of a binary classifier to differentiate between normal and abnormal classes. The ROC curve for the tested algorithms is plotted in Figure 2. AUC score 1 means a perfect classification whereas 0.5 value implies an insignificant result. ROC curves are generally used in the binary classification problem, to adapt ROC to the multi-classification problem we use macro-averaging method, which assigns equivalent weight to each class [37]. The best macro-averaged AUC is achieved by the RF classifier with 0.99 while KNN and SVM are reached at 0.95 and 0.97, respectively.

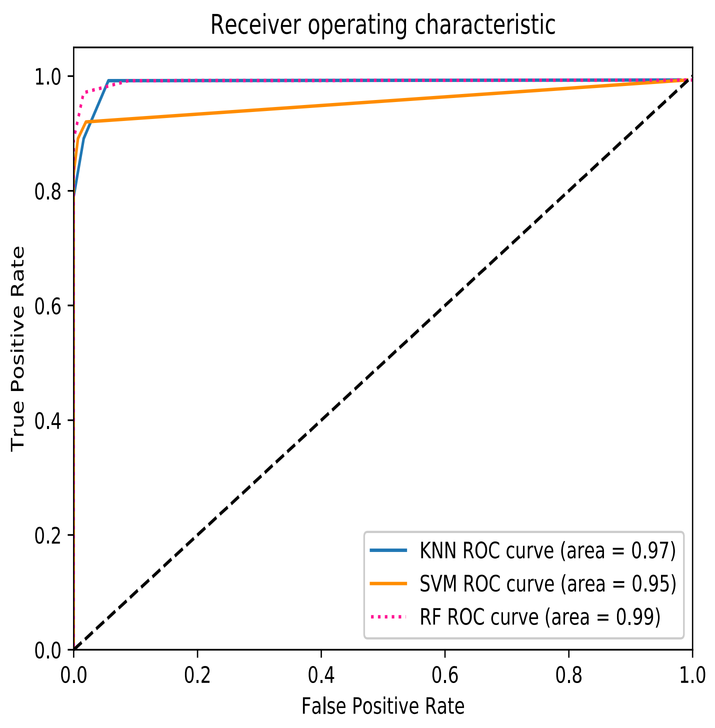


Figure 2. Macro-averaged ROC curves for tested ML algorithms.

In order to statistically compare two classifiers, 10-fold cross-validated paired t-test approach [38] is applied. When compared with RF and SVM algorithms using paired t-test, we conclude that RF performs significantly better than SVM since t-value (t = 3.275) is greater than p-value (p = 0.010). However, after applying paired t-test to RF and KNN, two classifiers perform equally well, since the p-value (p = 0.001) is smaller than the significance level of α = 0.005.

4. CONCLUSIONS

In this paper, we propose a novel sequential API pattern-based malware detection method. To this end, we dynamically analyze malware samples to extract API calls and then, we use sequential pattern mining algorithm to find API call patterns. Finally, the three machine learning algorithms are applied on API call patterns' features. Random Forest outperforms the other tested algorithms and achieved 99% F-measure. Experimental results on a fairly large-scale dataset show that the proposed approach reliably captures malicious representative and distinctive API call patterns and based on these patterns it can detect malware in an efficient and effective manner. Our proposed API call pattern-based malware detection approach is usable by many cyber-security tools to mitigate malware threats. Testing malware detection system with an extended the dataset is underway.

REFERENCES

[1] Statcounter: Operating system market share worldwide, (2018). http://gs.statcounter.com/os- market-share#monthly-201801-201801-bar. [Online; accessed 7-October-2017].

[2] Ilsun You & Kangbin Yim (2010) “Malware obfuscation techniques: A brief survey”, Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on, pp297– 300.

[3] 2016 Symantec Security Report, Internet: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf, 29.06.2018.

[4] Abdurrahman Pektas & Tankut Acarman (2018) “Malware classification based on api calls and behavior analysis”, IET Information Security, Vol. 12, No.2, pp 107-117.

[5] Abdurrahman Pektas¸ & Tankut Acarman (2014) “A dynamic malware analyzer against virtual machine aware malicious software”, Security and Communication Networks, Vol. 7, No.12, pp2245–2257.

[6] Nizar R Mabroukeh & Christie I Ezeife (2010) “A taxonomy of sequential pattern mining algorithms”, ACM Computing Surveys (CSUR), Vol. 43, No.1:3.

[7] Philippe Fournier-Viger & Jerry Chun-Wei Lin & Rage Uday Kiran & Yun Sing Koh & Rincy Thomas (2017) “A survey of sequential pattern mining”, Data Science and Pattern Recognition, Vol. 1, No.1, pp54–77.

[8] Yong Qiao & Jie He & Yuexiang Yang & Lin Ji (2013) “Analyzing malware by abstracting the frequent itemsets in api call sequences”,Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on, pp.265–270.

[9] Youngjoon Ki & Eunjin Kim & Huy Kang Kim (2015) “A novel approach to detect malware based on api call sequence analysis”, International Journal of Distributed Sensor Networks, Vol. 11, No.6, pp:95-10.

[10] In Kyeom Cho & Eul Gyu Im (2015), “Extracting representative api patterns of malware families using multiple sequence alignments”, In Proceedings of the 2015 Conference on research in adaptive and convergent systems, pp.308–313.

[11] Winfried Just (2001) “Computational complexity of multiple sequence alignment with sp-score”, Journal of computational biology, Vol. 8, No. 6. pp. 615–623.

[12] Lusheng Wang & Tao Jiang (1994), “On the complexity of multiple sequence alignment”, Journal of computational biology, Vol. 1, No.4, p.337–348.

[13] Yujie Fan &Yanfang Ye & Lifei Chen (2016), “Malicious sequential pattern mining for automatic malware detection”, Expert Systems with Applications, Vol.52, pp.16–25.

[14] Iltaek Kwon & Eul Gyu Im (2017), “Extracting the representative api call patterns of malware families using recurrent neural network”, In Proceedings of the International Conference on Research in Adaptive and Convergent Systems, pp.202–207.

[15] Canfora, G., Mercaldo, F., & Visaggio, C. A. (2016). An hmm and structural entropy based detector for android malware: An empirical study. Computers & Security, 61, 1-18.

[16] Salehi, Z., Sami, A., & Ghiasi, M. (2017). MAAR: Robust features to detect malicious activity based on API calls, their arguments and return values. Engineering Applications of Artificial Intelligence, 59, 93-102.

[17] Shijo, P. V., & Salim, A. (2015). Integrated static and dynamic analysis for malware detection. Procedia Computer Science, 46, 804-811.

[18] Cuckoo Sandbox, Internet: https://cuckoosandbox.org/, 29.06.2018.

[19] Virustotal, Internet: https://www.virustotal.com/, 29.06.2018.

[20] Payam Refaeilzadeh & Lei Tang & Huan Liu (2009) "Cross-validation", In Encyclopedia of database systems, pp.532–538, Springer.

[21] A. Barthels, Behavior-based Malware Detection, Faculty of Informatics, The Technical University of Munich, Master Thesis, 2009.

[22] Chand, C., Thakkar, A., & Ganatra, A. (2012). Sequential pattern mining: Survey and current research challenges. International Journal of Soft Computing and Engineering, 2(1), 185-193.

[23] Parikh, M., Chaudhari, B., & Chand, C. (2013). A comparative study of sequential pattern mining algorithms. International Journal of Application or Innovation in Engineering & Management (IJAIEM), 2(2).

[24] Mooney, C. H., & Roddick, J. F. (2013). Sequential pattern mining--approaches and algorithms. ACM Computing Surveys (CSUR), 45(2), 19.

[25] Ramakrishnan Srikant & Rakesh Agrawal (1996), "Mining sequential patterns: Generalizations and performance improvements", In International Conference on Extending Database Technology, pp.1–17, Springer.

[26] Jay Ayres & Jason Flannick & Johannes Gehrke & Tomi Yiu (2002) "Sequential pattern mining using a bitmap representation", In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp.429–435.

[27] Mohammed J Zaki. Spade (2001) "An efficient algorithm for mining frequent sequences. Machine learning", Vol.42, No.1-2, pp.31–60.

[28] Philippe Fournier-Viger &Antonio Gomariz & Ted Gueniche &Azadeh Soltani & Cheng-Wei Wu & Vincent S Tseng (2014) "Spmf: a java open-source pattern mining library", The Journal of Machine Learning Research, Vol.15, No.1, pp.3389–3393.

[29] SPMF library, Internet: http://www.philippe-fournier-viger.com/spmf/, 29.06.2018.

[30] Philippe Fournier-Viger & Antonio Gomariz & Manuel Campos & Rincy Thomas (2014) "Fast vertical mining of sequential patterns using co-occurrence information", In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp.40–52, Springer.

[31] Gandotra, E., Bansal, D., & Sofat, S. (2014). Malware analysis and classification: A survey. Journal of Information Security, 5(02), 56.

[32] Leo Breiman (2001) "Random forests", Machine learning, Vol.45, No.1, pp.5–32.

[33] Padraig Cunningham & Sarah Jane Delany (2007) "k-nearest neighbour classifiers", Multiple Classifier Systems, Vol.34, pp.1–17.

[34] Marti A. Hearst & Susan T Dumais & Edgar Osuna & John Platt & Bernhard Scholkopf (1998), "Support vector machines", IEEE Intelligent Systems and their applications, Vol. 13, No.4, pp.18–28.

[35] Fabian Pedregosa & Gaël Varoquaux &Alexandre Gramfort & Vincent Michel & Bertrand Thirion & Olivier Grisel & Mathieu Blondel & Peter Prettenhofer &Ron Weiss &Vincent Dubourg (2011)

"Scikit-learn: Machine learning in python", Journal of machine learning research, Vol. 12, pp.2825–2830.

[36] Hossin, M., & Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. International Journal of Data Mining & Knowledge Management Process, 5(2), 1.

[37] Yiming Yang (1999) "An evaluation of statistical approaches to text categorization", Information retrieval, Vol.1, No. 1-2, pp.69–90.

[38] Thomas G Dietterich (1998), "Approximate statistical tests for comparing supervised classification learning algorithms", Neural computation, Vol.10, No.7, pp.1895–1923.

AUTHORS

Abdurrahman Pektaş received his B.Sc. and M Sc. at Galatasaray University and his PhD at the University of Joseph Fourier, all in computer engineering, in 2009, 2012 and 2015, respectively. He is a senior researcher at Galatasaray University. His research interests are analysis, detection and classification of malicious software, machine learning and security analysis tool development.

Elif Nurdan Pektaş received his B.Sc. and M Sc. at Galatasaray University all in computer engineering, in 2010, and 2014, respectively. She is leading software developer at Siemens Turkey. Her research interests are developing IoT based applications, deep learning, cloud based application and automated testing.

Tankut Acarman received his Ph.D. degree in Electrical and Computer engineering from the Ohio State University in 2002. He is professor and head of computer engineering department at Galatasaray University in Istanbul, Turkey. His research interests lie along all aspects of autonomous systems, intelligent vehicle technologies and security. He is the co-author of the book entitled "Autonomous Ground.