Received: March 1, 2017

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

ESAA: Efficient Sequence Alignment Algorithm for Dynamic Malware Analysis in Windows Executable Using API Call Sequence

Asha Jerlin[1]* **Jayakumar Chinnappan[2]**

[1]*Vellore Institute of Technolog University,Vellore, India*
[2]*Sri Venkateswara College of Engineering ,Sriperumputhur, India*
* Corresponding author's Email: ashajerlin.m@vit.ac.in

Abstract: Detection of malware has become more challenging today because of the advancements and technologies adapted to corrupt the network or the devices. Static, dynamic and hybrid malware detection analysis methods have failed to provide complete malware detection. Hence in this work, a bio inspired sequence alignment method used in bioinformatics to compare the similarity between amino acid sequences in protein structures has been adapted to give the best similarity score to detect malwares. The state of art sequence alignment methods like Smith Water Man Algorithm (SWMA) used in bio informatics suffers from the problem of more memory utilization and computation time which is in the order of n2 ie., (O(n2)) and hence in this work an efficient sequence alignment algorithm (ESSA) has been proposed to address the problem of memory utilization thereby making the memory utilization and computation time to the order of n ie., (O(n)) there by making the detection rate higher. It is also clear from the results that the similarity score is high when the sequence length is small. The accuracy of the prediction rate of malware and benign increases.

Keywords: Malware analysis, Sequence alignment, Memory utilization, Similarity, benign.

1. Introduction

If we get in to the history of malware analysis techniques, it started with static analysis then dynamic analysis and some researchers have worked on hybrid techniques also. Due to the influence of bio inspired methods and its similarity in analysing the DNA sequence, it has been deployed in the computer malware analysis and proved to be an interesting and an effective method.

The McAfee labs report [1] on 2016 Threat predictions claims that behavioural analysis is the best suited approach for detecting the attacks. Unfortunately, there seems to be no solid analysis technologies available to gain the upper hand .It is predicted to take another five years. Behavioural / dynamic analysis is the next big weapon in the security defence tool kit.

Behavioural analytics are still in its early stages and being challenging to extract meaningful information from massive dataset and will definitely mature quickly in the next five years if the skills in machine learning and analytics address the problem [1].

Static analysis: Malware detection at early stages used static analysis [2-3]. Static analysis identifies the malicious code without execution of binary codes [4].Hackers are clever enough to make the binary code analysis very difficult by emerging with more sophisticated techniques of obfuscation, polymorphism, encryption and packing [5-6].

Dynamic analysis: In dynamic analysis the benign and the malware executable are executed in a packed virtual environment called sandbox and there by their behaviour is traced. Dynamic analysis can be done using two techniques: Control flow graph analysis and API call sequence analysis. Researches claim that there is a research Gap in dynamic analysis since it is not accurate as static analysis and hence the researchers have ample challenges to reduce the False Positive Rate (FPR) thereby increasing the accuracy [6-7].

DNA sequence Alignment: The DNA sequence alignment algorithm provides a solution to match the maximum common sequence of two target sequences. They are found to provide better accuracy than other methods [8]. The research gap available with this method is the use of memoization techniques for sequence alignment which suffers from space complexity.

The organization of the report is as follows:
In section 2 the survey of the related work is discussed, section 3 explains the proposed work and the methodology in detail, section 4 explains the proposed ESAA method with its working and calculation, sections 5&6 concludes the paper with experimental results.

2. Literature survey

Natani et al [8] proposed a method for the malware detection by using the API frequencies and ensemble based classifiers. In their work, authors had used multi-classifiers instead of a single classifier to analyze malware. They analyzed 100 malicious behaviors and choose 24 APIs used for that malicious behaviors. Sand box were used to measure frequency of API. Finally the accuracy was measured.

Wagener et al. [9] extracted the behavior information of malware by observing that malware invoked the system functions. Then, they compared the malware' API invocation information and calculated similarities among malware variants. Their proposed method defined binary code for each API. The binary code was used to store API invocation patterns of malware. For similarity calculation, they used an edit distance matrix, and also used their own formula. Finally they calculated the similarities of malware variants.

Xu et al [10] proposed the approach for the various malwares detection in the Windows platform. They extracted the API call sequences by analyzing PE binary code and calculated the similarities of the API sequences between the unknown malware and the known one. They implemented a PE binary parser tool themselves, because the third-party disassemblers extract some unnecessary features of malware samples and these unnecessary features made the performance of the analysis system very low. This tool generates the API sequences of malware, and if the malware are known, the sequences were stored in asignature database system. Those sequences are compared with the API sequences of unknown malware, then the similarity of the two sequences is calculated.

Liu Wu et al [11] investigated techniques of malware behavior extraction based malicious API invocation, and presented the formal Malware Behavior Feature (MBF) extraction method. This Malware Behavior Feature was expressed in Boolean, and they proposed the malicious behavior feature based malware detection algorithm. Finally, they designed and implemented the MBF based malware detection system, and the experimental results showed that the accuracy rate of their MBF based detection system is high and it can detect newly appeared unknown malware.

Martin Apel et al [12] investigated distance metrics to detect polymorphic malware effectively. Distance metrics are different distance measures in detail and they discussed desirable properties of a distance measure. They focused on behavioral features of malware and compared and experimentally evaluated different distance measures for malware behaviors. They selected an appropriate distance measure for grouping malware samples based on similar behaviors.

Youngjoon Ki et al [13] in their work used the methods of DNA cluster alignment, MSA and used API and control flow as features to detect malware based on similarity. Randomly taken 23,080 malware samples from the malware dataset of malicia project and virus total were used as dataset. In this dataset 2727 kinds of API's were used. Their method was able to overcome the need of time and memory of MSA. Kernel level API's were not addressed.

Eva Czabarka et al [14] had formulated their work using Chain termination method. This model allows determining how much sequencing of shortgun to be performed. The dynamic programming algorithm has been used. Their results show that Automated DNA sequencing and the analysis lead to efficiency gain.

Kyeom Cho et al [15] had discussed in their work about MSA which is used to measure similarities among multiple malware variants. Sandbox tool is used to extract API call sequences of malware. Sandbox and modified clustal omega is used for experiments. Samples- 15 malware families are collected from VxHeaven. This system measures classification accuracy of API patterns of each malware families using clustering technique, if API call sequence of malware sample is long process then MSA is not suitable.

Kyeom Cho et al [16] in their work used Sequence alignment method -n-gram-MBF extraction method-Local alignment method-Global alignment method-Malware similarity calculation method. Sequence alignment method detects similarity in parts of the malware's API call sequences-Malware variants have different structural static features like mnemonic frequencies controls flow graphs.Malware samples from VxHeaven.150 malware samples from 10 malware families,each family has 15 malware variants.

Piero Fariselli et al [17] in their paper has describe a general dynamic programming-like algorithm (MaxSubSeq, Maximal SubSequence) specifically designed to optimize the number and length of segments with constrained length in a given protein sequence.

Vivek Kumar et al [18] in their work had used Multiple Sequence analysis, Needleman-Wunsch, Smith waterman technique to conservation of variable length of biological sequences converted to fixed length sequences by inserting and deleting the gaps. Their tests fail with large amount of dataset.

Yi Chen et al [19] proposed an experiment of converting the hexadecimal code of viruses and worms to amino acids and ASCII form are more effective and also separating worms from viruses. Further work is required to ensure that the insertion of gaps does not lead to such gaps being included in the rules for producing signatures.

3. Proposed work

In the proposed work, the detection of malware using sequence alignment given in Fig.1 is as follows: [20]

a) Collection of malware and benign samples through dynamic execution

b) API call sequence retrieval from the dynamic behaviour

c) N-gram sequence feature extraction

d) Application of ESAA algorithm to the n-gram sequence

e) Testing the classifiers for accuracy with the test data

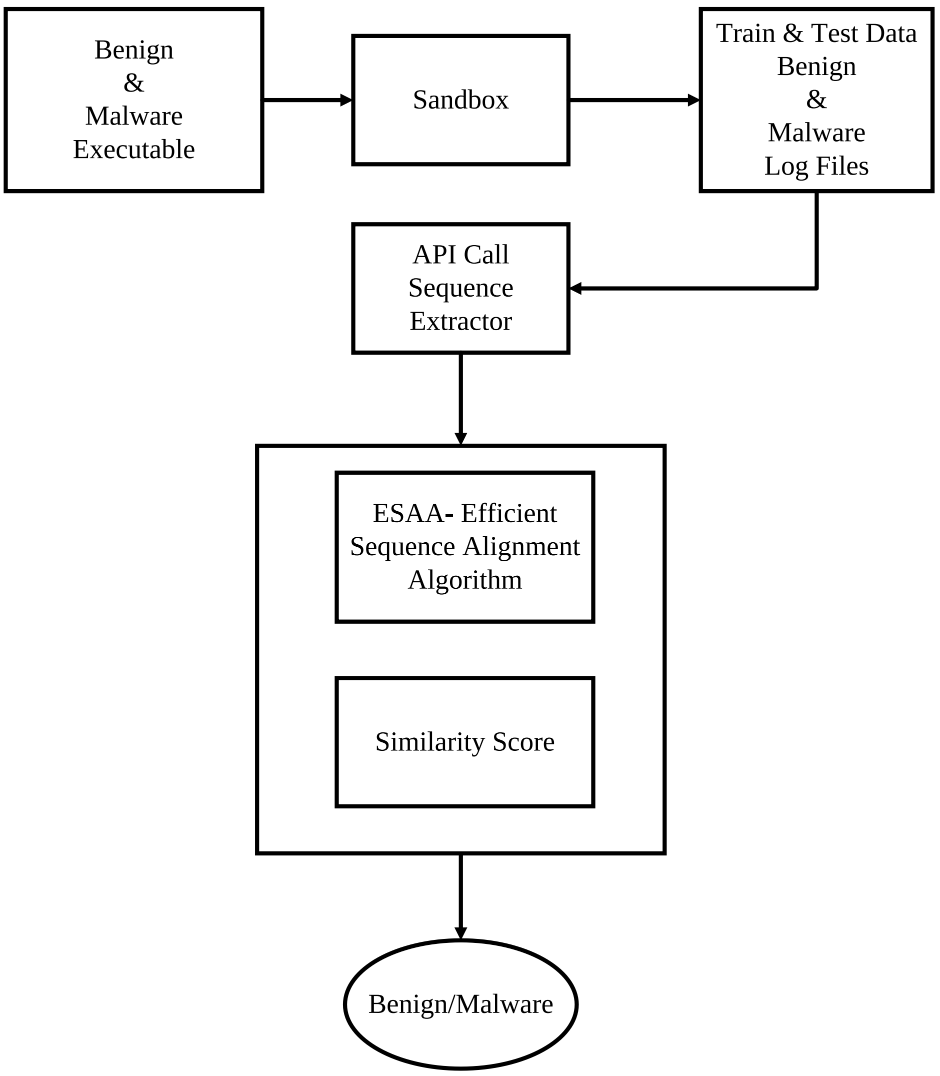
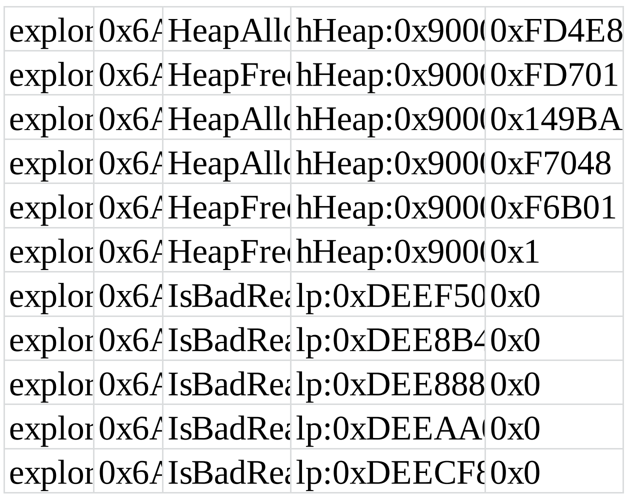


Figure.1 Architecture of malware analysis

Received: March 1, 2017



explor	0x6	HeapAll	hHeap:0x900	0xFD4E8
explor	0x6	HeapFre	hHeap:0x900	0xFD701
explor	0x6	HeapAll	hHeap:0x900	0x149BA
explor	0x6	HeapAll	hHeap:0x900	0xF7048
explor	0x6	HeapFre	hHeap:0x900	0xF6B01
explor	0x6	HeapFre	hHeap:0x900	0x1
explor	0x6	IsBadRe	lp:0xDEEF50	0x0
explor	0x6	IsBadRe	lp:0xDEE8B	0x0
explor	0x6	IsBadRe	lp:0xDEE888	0x0
explor	0x6	IsBadRe	lp:0xDEEAA	0x0
explor	0x6	IsBadRe	lp:0xDEECF	0x0

Figure.2 Malware Log File (Excel Sheet containing API Call sequence data)

File name	N-gram	Frequency
XTrojan.Win32. Alcapul	RegOpenKeyExW GetProcAddress HeapAlloc	15

Figure.3 Map data structure

3.1 Methodology [20]

Dataset creation: 1696 malware executable was collected from www.virusshare.com[20] and 1042 benign files were collected from the windows applications tested under McAfee anti-virus. The dataset was created by executing the malwares and logging the behaviour it by executing it in a virtual environment [20] using cuckoo sandbox [20].

3.2 API call sequence extraction [20]

The API call sequences were used as features by extracting it from the logs.

3.3 Feature selection and data representation [20]

The n-gram call sequence is extracted from the API. In this work, 3-grams & 4-grams were extracted for analysis. The data representation is made as a feature set with 645 columns and 482. The columns indicate 3- gram & 4- grams and the column indicate the classes.

3.4 Classification by sequence alignment [20]

The ESAA algorithm is applied on the extracted 2-gram & 3-gram API call sequence in order to perform the classification by sequence alignment.

3.5 Architecture

The data collected from source[20] is given to a malware analyzer for the creation of log files and then the API call sequence is extracted from the executable. With the available API N-grams are extracted to form data representation from where the dynamic feature vector is created and training data is given for classification using ESAA .

3.5.1. Malware Analyser [20]

An Executable File is given as the input to this module in order to classify it as either Benign or Malware, this could be done by first gathering the behavioural data of that file. The Behavioural data is collected by dynamically executing the file in a secured environment called sandbox environment. The Malware Analyser module interprets the Hexadecimal code format got from the executable to API Call Sequence data shown in Fig.2.

3.5.2. API Call Sequence Extractor [20]

The excel file containing API Call sequence data is given as an input to this module.The process of creating a data structure is done here.The API Call sequence data presented by the malware analyser module in the form of excel sheet is read using Java Apache POI library and a data Structure is created. The data Structure used to Store API Call Sequence Data is a HashMap. The final output of this module is API Call Sequence.

3.5.3. N-Gram Extractor [20]

This module takes the data Structure obtained from the extractor module containing API Call Sequence Data. The process of N-Grams creation is done and stored in Map Data structure. The output of the module is a map containing File name, its related n-grams and frequency of the n-gram as shown in Fig.3.

3.5.4. Feature Selection and Feature Creation [20]

The 3-Gram and 4-Gram API call presented in Map Data structure is given as an input to this module .The feature Vector is created ,which is the combination of 0’s and 1’s. If the 3-gram or 4-gram is present in the particular file then it is marked as 1 else 0. The output obtained from this module is an excel file containing feature vector as shown in Fig. 4.

Hea	Get(	Hea	Hea	Get(	Get(	Hea	IsBa	Get(	Hea	Get(	Loca	Hea
0	1	1	1	1	1	0	1	0	1	1	1	1
0	1	1	1	0	1	0	0	0	1	1	1	1
0	1	1	1	0	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1	1	1
0	1	1	1	0	1	0	1	0	1	1	1	1
0	1	1	1	0	1	0	1	0	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1	1	1

Figure.4 Feature Vector

3.5.5. Similarity calculation

The feature vector obtained from the previous module is given as input for similarity calculation. The Feature vector is provided to the Efficient Sequence Alignment Algorithm (ESAA) for finding the score and the similarity value for classifying the input executable as Benign or Malware.

Similarity= Maximum score / Minimum sequence length

4. Efficient sequence Alignment Algorithm (ESAA)

```
Input: string s1 of length m, string s2 of length n

1. The API features of the raw dataset are compared with the other features in the reduced feature vector called RFList.

2. Two single dimension arrays are declared.

    2.1 First column in the matrix is taken as previous row that is first array

    2.2 Current row values are calculated in the second array.

3. Initially, the first row and column values are marked zero.

    3.1 Let i==1

    3.2 Declare an array prevRow[], Let x=0 and x<seq2.length prevRow[x]=0;

    3.3 Declare an array currRow[];
        currRow[0]=0;

4. Compute the left value, up value and diagonal value of each cell using

    leftVal=currRow[j-1];

    upVal=prevRow[j];

    diagVal=prevRow[j-1];

5. If the sequence 1 and the sequence 2 taken for matching are equal then

    temp = max(leftVal+GAP,upVal+GAP,diagVal+MATCH); else

    temp=max(leftVal+GAP,upVal+GAP,diagVal+MIS_MATCH);

6. If temp value computed in the previous step is less than zero then current row value is marked zero otherwise the temp value computed is take as the current row value.

    currRow[j] = temp<0 ? 0 : temp

7. Compute the maximum score value using

    if(temp>max_score)

    max_score=temp;

8. Output: value of optimal alignment
```

4.1 Working of ESAA

Consider an alignment of two sequences S and T. The computation is based on matrix say M. Create an NxN integer matrix. N is sequence length (both S and T). Compute M[i][j] based on score matrix and optimum score. Consider the following two sequences:

S: c c c t a g g t c c c a

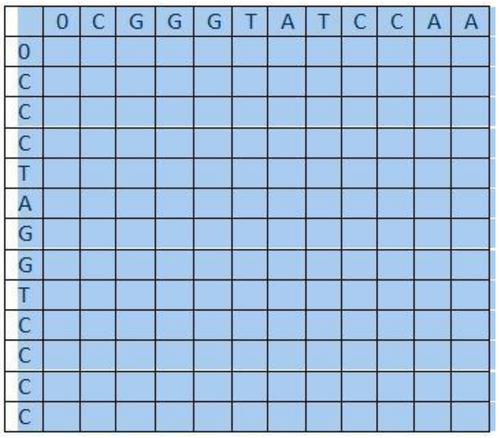
T: c g g g t a t c c a a

Step 1: Computing matrix alignment, M is created as follows in Fig.5 by taking the string elements S in columns and T in rows.

Step 2: Aligning S to gaps. The first values of the column are marked 0 by default as shown in Fig.6.

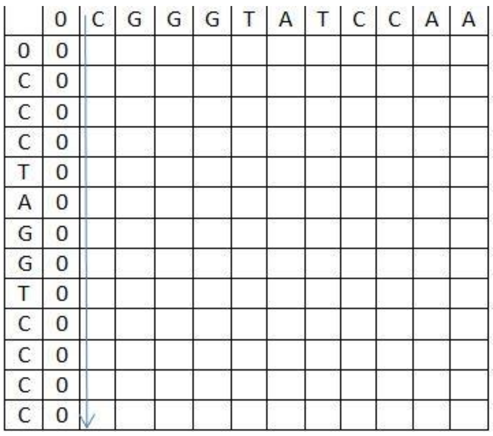
T: - - - - - - - - - - - -

S: c c c t a g g t c c c a



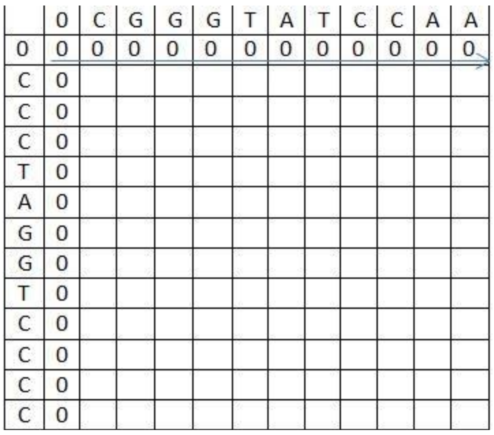
	0	C	G	G	G	T	A	T	C	C	A	A
0												
C												
C												
C												
T												
A												
G												
G												
T												
C												
C												
C												
C												

Figure.5 Initial matrix alignment



	0	C	G	G	G	T	A	T	C	C	A	A
0	0											
C	0											
C	0											
C	0											
T	0											
A	0											
G	0											
G	0											
T	0											
C	0											
C	0											
C	0											
C	0											

Figure.6 Filling first column with zero



	0	C	G	G	G	T	A	T	C	C	A	A
0	0	0	0	0	0	0	0	0	0	0	0	0
C	0											
C	0											
C	0											
T	0											
A	0											
G	0											
G	0											
T	0											
C	0											
C	0											
C	0											
C	0											

Figure.7 Filling first row with zero

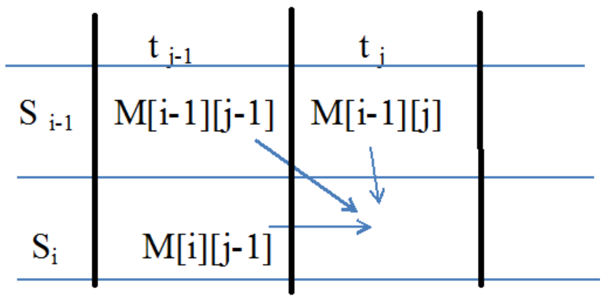


Figure.8 Matrix Calculation

Step 3: Aligning T to gaps: All the first values of the row are also marked 0 by default as shown in Fig.7.

T: c g g g t a t c c a a

S: - - - - - - - - - - - -

Step 4: The score for each element in the cell of the matrix is computed by a scoring process as shown in Fig.8.

4.2 Calculations

Step 1: To find a cell value in the matrix M[i][j], the following initialization is made

M[i][0]= 0 M[0][i]= 0;

GAP= -1, MATCH= 1, MISMATCH = -1 (1)

Step 2: Each cell value is found by calculating the (i) upper cell value+ Gap, (ii)left cell value +Gap, (iii)Diagonal cell value +Match/ Mismatch. While calculating Match /Mismatch compare the row & column variable.

Step 3: If the final value obtained for the call is less or negative then it is considered to be '0'.

Step 4: The final matrix created by finding the value for each cell using Eqs.(1), (2) and step 2 & 3 gives the maximum score in it.

Step 5: Applying the non-memoization technique.

$$M[i][j] = \max \begin{Bmatrix} M[i-1][j-1] + S[s_i][t_j] \rightarrow s_i t_j \\ M[i-1][j] \rightarrow s_i \\ M[i-1][j-1] - d \rightarrow t_j \end{Bmatrix} \quad (2)$$

4.3 Memoization & Non-memoization technique

Memoization Technique

- Consumes Huge RAM.
- Cannot be applicable for Millions of records.

Ex: 100000 * 100000 rows = 10000000000 * 4 bytes = 40000000000/(10^9) = 40GB RAM!!!

Table 1. Score and similarity values

S.No	File Category	Score	Similarity
1	Xaddress book.xlsx	12438	0.99
2	Xcalculator.xlsx	5051	0.42
3	XTrojan-Spy.Win32.Delf.di.apm.xlsx	1419	0.11
4	XVirus.Win32.BingHe-ee.apm.xlsx	294	0.22
5	XWorm.Win32.Bumerang-ee.apm.xlsx	290	0.04

Table 2. Sample output run

```
run:
C:\Users\Dharaneesh N\Documents\dataset-used\both
Malware and Benign
File Name Xaddress book.xlsx Score 12438 Similarity
0.9999196076855053
File Name Xadobe distiller.xlsx Score 2199 Similarity
0.17678269957392073
File Name Xalcohol.txt.xlsx Score 2581 Similarity
0.24769673704414588
File Name Xbackgammon.xlsx Score 5451 Similarity
0.47466039707419017
File Name Xburstcopy.txt.xlsx Score 2531 Similarity
0.36777099680325487
File Name Xcalculator.xlsx Score 5051 Similarity
0.42563411140136514
File Name Xcheckers.xlsx Score 5233 Similarity
0.4477624711217592
File Name Xclone dvd.xlsx Score 4131 Similarity
0.3818988629009892
File Name Xdaemon.xlsx Score 4663 Similarity
0.5379556991232118
File Name Xdap.xlsx Score 4317 Similarity
0.34705362167376796
File Name Xdataobjectviewer.xlsx Score 5379 Similarity
0.5493259803921569
File Name Xddespy.xlsx Score 4545 Similarity
0.3653830693785674
File Name Xdeviceemulator.xlsx Score 354 Similarity
0.028458879331135944
File Name Xdigsby.xlsx Score 3952 Similarity
0.31771042688318996
File Name Xdiskdefragment.xlsx Score 5545 Similarity
0.44577538387330173
File Name Xdjvu reader.xlsx Score 2930 Similarity
0.2355494814695715
File Name Xdvi viewer.xlsx Score 5221 Similarity
0.4197282739770078
File Name Xeainfo.xlsx Score 4740 Similarity
0.3810595707050406
File Name Xeareg.xlsx Score 4772 Similarity
0.3836321247688721
```

- The aligned sequence can be retained
- Space Complexity – O(n^2)

Non–Memoization Technique

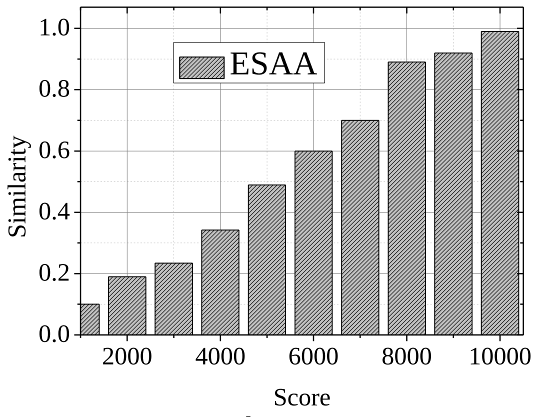


Figure.9 Similarity vs. score

- Just remember the current and the previous row.
- Keeps track of the maximum similarity score.
- Space Complexity – O(n)
- Can be applicable for Millions of records too.
- The aligned sequence cannot be retained

5. Results and Discussion

An 'n' number of runs were made to find the maximum score and the similarity of each executable to distinguish it as malware or benign. The below table 1 gives the sample output of the run. The similarity index of benign files are found to be greater than 0.4. The similarity is calculated using Eq. (3)

$$\text{Similarity} = \text{Maximum score} / \text{Minimum sequence length} \qquad (3)$$

5.1 Similarity vs. score value

From Fig.9, it is clear that as the score value increases the similarity of the sequence also increases making the prediction rate and accuracy high. Table 2 denotes the sample output of the run.

5.2 Memoization vs. non memoization technique

The existing sequence alignment algorithm called Smith Water Man Algorithm(SWMA) uses a technique called memorization technique takes more memory for creating the matrix. For instance,

• Ex: 100000 * 100000 rows = 10000000000 * 4 bytes = 40000000000/(10^9) = 40GB RAM!

Received: March 1, 2017

which requires a multidimensional storage and thereby making the memory utilization as O(n^2).
The proposed Efficient Sequence Alignment Algorithm (ESAA) method uses the non-memoization technique which considers only the current row and column value for calculating the current cell value and hence uses two one dimensional arrays for storage thereby decreasing the memory utilization by O(n). This can be viewed in Fig.10.

5.3 Similarity Vs. Length of sequence

The length of the sequence plays a vital role in the similarity calculation. When the length increases similarity decreases. Hence the proposed algorithm decreases the length of the sequence is reduced by the gap values which is depicted in Fig.11.

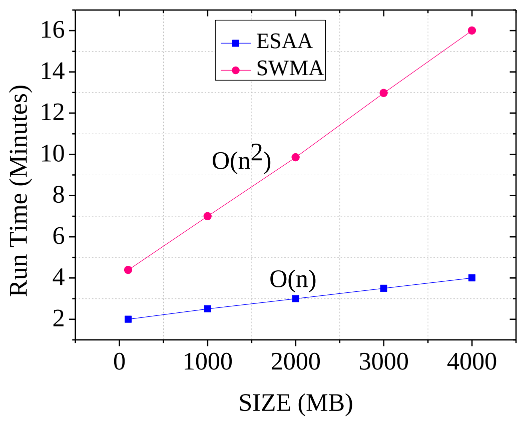


Figure.10 ESAA vs SWMA

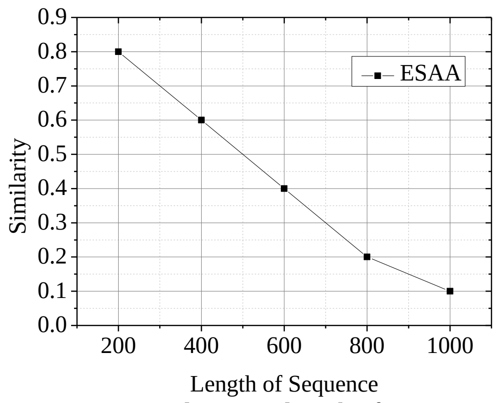


Figure.11 Similarity vs. length of sequence

6. Conclusion & Future Work

From the experimental results it is concluded that the proposed ESAA method used in this work for sequence alignment reduces the memory utilization. The existing sequence alignment methods shows a memory utilization by order of n^2 [O(n^2)],whereas the proposed method reduced the memory utilization by order of n [O(n)]. Also the length of the sequence is maintained by the gap values so that the similarity increases there by increasing the score value which will influence the prediction accuracy.

The main limitation of this proposed ESAA method is it cannot backtrack since the previous values cannot be retained. As a future work the algorithm can be modified to perform backtracking of the cell values and to retain the previously aligned sequence there by not compromising on the memory utilization.

References

[1] McAfee Labs Report 2016 Threats Predictions, www.mcafee.com/in/resources/reports/rp-threats-predictions-2016.pdf.

[2] N. Idika, and A. Mathur, "A survey of malware detection techniques", *Purdue University, West Lafayette*, Vol.48, 2007.

[3] K. Han, I.K. Kim, and E.G. Im, "Malware classification methods using API sequence characteristics", In: *Prof. of the Conference on IT Convergence and Security, Springer, Netherlands*, pp. 613-626, 2012.

[4] P. O'Kane, S. Sezer, and K. McLaughlin, "Obfuscation: The hidden malware", *IEEE Security & Privacy*, Vol.9, No.5, pp.41-47, 2011.

[5] V. Sathyanarayan, P. Kohli, and B. Bruhadeshwar, "Signature generation and detection of malware families". In: *Proc. of the Australasian Conference on Information Security and Privacy, Springer,Berlin Heidelberg*, pp. 336-349, 2008.

[6] R. Tian, R. Islam, L. Batten, and S. Versteeg, "Differentiating malware from cleanware using behavioural analysis", In: *Proc. of the 5th International Conference on Malicious and Unwanted Software, IEEE*, pp.23-30, 2010.

[7] A. Jerlin, and C. Jayakumar, "A dynamic malware analysis for windows platform-a survey", *Indian Journal of Science and Technology*, Vol. 8, No.27, pp. 1-5, 2015.

[8] P. Natani, and D. Vidyarthi, "Malware detection using API function frequency with ensemble based classifier", In: *Proc. of the International Symposium on Security in Computing and Communication, Springer, Berlin Heidelberg*, pp. 378-388, 2013.

[9] G. Wagener and A. Dulaunoy, “Malware behaviour analysis”, *Journal in computer virology,* Vol.4, No.4, pp.279-287, 2008.

[10] J. Xu, A. Sung, P. Chavez, and S. Mukkamala, “Polymorphic malicious executable scanner by API sequence analysis”, In: *Proc. of the Fourth International Conference on Hybrid Intelligent Systems, IEEE,* pp. 378-383, 2004.

[11] W. Liu, P. Ren, K. Liu, and H. Duan, “Behavior-based malware analysis and detection”, In: *Proc. of the First International Workshop on Complexity and Data Mining, IEEE,* pp. 39-42, 2011.

[12] M. Apel, C. Bockermann, and M. Meier, “Measuring similarity of malware behaviour”, In*: Proc. of the IEEE 34th Conference on Local Computer Networks, IEEE,* pp. 891- 898, 2009.

[13] Y. Ki, E. Kim, and H. Kim, “A novel approach to detect malware based on API call sequence analysis”, *International Journal of Distributed Sensor Networks,* Vol.2015, No.1, pp.1-9, 2015.

[14] E. Czabarka, G. Konjevod, M. Marathe, A. Percus, and D. Torney, “Algorithms for optimizing production DNA sequencing”, *In: Proc. of the ACM-SIAM symposium on Discrete algorithms,* pp. 399-408, 2000.

[15] I.K. Cho and E.G. Im, “Extracting representative API patterns of malware families using multiple sequence alignments”, In: *Proc. of the Conference on research in adaptive and convergent systems,ACM,* pp. 308-313, 2015.

[16] I.K. Cho, T. Kim, Y.J. Shim, H. Park, B. Choi, and E.G. Im, “Malware similarity analysis using API sequence alignment”, *Journal of Internet Services and Information Security,* Vol.4, No.4, pp.103-114, 2014.

[17] P. Fariselli, M. Finelli, D. Marchignoli, P. Martelli, I. Rossi, and R. Casadio, “MaxSubSeq: an algorithm for segment-length optimization. The case study of the transmembrane spanning segments”, *Bioinformatics,* Vol.19, No.4, pp.500-505, 2003.

[18] V. Kumar and S. Mishra, “Detection of malware by using sequence alignment strategy and data mining techniques”, *International Journal of Computer Applications,* Vol.62, No.22, pp.16-19, 2013.

[19] Y. Chen, A. Narayanan, S. Pang, and B. Tao, “Malicious software detection using multiple sequence alignment and data mining”, In: *Proc. of the IEEE 26th International Conference on Advanced Information Networking and Applications,* pp. 8-14, 2012.

[20] A. Jerlin, C. Jayakumar, and J. Prabhu, “EFE: Efficient Feature Extraction Algorithm for dynamic malware analysis in windows executables using API call sequence”, *International Journal of Pharmacy & Technology,* Vol.8, No.4, pp.25373-25383, 2016.