Journal of Theoretical and Applied Information Technology
31st May 2017. Vol.95. No 10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ESMP: EXPLORATORY SCALE FOR MALWARE PERCEPTION THROUGH API CALL SEQUENCE LEARNING

[1]G. BALAKRISHNA, [2]DR. V.RADHA, [3]DR. K.VENU GOPALA RAO

[1]Associate Professor, Department of CSE, CVR College of Engineering, JNTUH, Hyderabad, INDIA

[2] Assistant Professor, IDRBT, Hyderabad, INDIA

[3] Professor, Department of CSE, GNITS, JNTUH, Hyderabad, INDIA

E-mail: [1] govind.krishna83@gmail.com,

ABSTRACT

One of the critical factor of computer aided services and data security is defending malicious executables known as malwares. Since the zero day activities of malware, it becomes continuous process to sense and prevent the malicious activities of the vulnerable executables. The contemporary literature evinces the many of malware detection approaches. The malware detection by dynamic assessment is figured as significant to explore the behavioral information of the malicious executables. The recent malware analysis is concluding that the act of obfuscating the malicious executables is boosting the complexity of defending such attacks. This practice strongly demanding the more accurate malware defending approaches, hence this manuscript contributed an exploratory scale to analyze API call sequence in order to estimate the scope of malicious act by an executable. The proposed model called Exploratory Scale for Malware Perception (ESMP) is a machine learning strategy that acquires knowledge from the defined executables that labeled as either malicious or benevolent. Further this knowledge is used to define the exploratory scale proposed. ESMP even capable of identifying zero day exploiting of malware. The experimental study was carried out on set of executables labeled as either malicious or benevolent. The 70% of the given executables were used to train the ESMP to define exploratory scale and rest 30% were unlabeled and given to test the significance of the ESMP towards malware detection accuracy. The statistical metrics such as accuracy, sensitivity and specificity were assessed to notify the scalability, robustness and detection accuracy of the ESMP.

Keywords: *Executables, Malwares, Benevolent, Zero day activities, Exploratory.*

1. INTRODUCTION

Over 20 years since malware first showed its presence, there has been a huge proliferation in malware numbers [1] [2] [3]. As world-wide internet access increased so did the exploitation of network vulnerabilities by malicious software. The malware multiplication is so huge that the year 2014 has seen more than 317 million new malware samples being detected. The different malware such as new viruses, worms, Trojans and bots are created by their authors to exploit and control the target network/system for explicit gains or with malicious intentions. The prime reason for the spread of a number of malware is the malware authors creating newer variations of the original malware using syntax code different from the existing malicious code. The newer malware are created with structural transformation that introduces variations of the existing static features such as mnemonic frequencies, data constants, control flow graphs, etc. However most of static analysis based malware detection systems are dependent on matching the signature of a newly identified malware with the signatures available of the existing database of malware [4] [5] [6] and this limitation is successfully exploited by the newer malware that have transformed code and go undetected. So the approach of using simple static analysis is inefficient in handling the threats posed by newer variants emerging every day. The traditional signature based anti-malware scanners can only detect known variants of malware and cannot protect against Zero day attacks and unseen

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

malware which is answered in this proposed research work.

The remainder of this paper is organized as follows: Section 2 provides the related work of the subject of the paper. Section 3 discusses the sequence alignment subject along with a problem description in implementing this for malware detection and analysis. In Section 4 we present our devised similarity calculation system for detecting variations in the malware and give a detailed explanation of the modules of the proposed framework. Section 5 demonstrates the results of the experiments that show the prospects of detecting variations in malware using variant samples along with other assessment results. In Section 6 a summary of our performed research is given discussing the proposed methodologies advantages and deficiencies and covers the future research work currently undertaken and its characteristics.

2. RELATED WORK

The research for malware detection has many methods devised and most of them describe malicious behaviors using features of Application Programming Interface (API) where the behavior of the program is captured at API level. The API calls approach assures detection of malware variants and these methods also have self-defined techniques for performing code comparison of the different variants of malware.

The ensemble based techniques have been a popular approach in machine learning where the algorithms are used with boosting or bagging techniques to enhance the accuracy and stability performances. The method devised by Natani et al. [7] for malware detection and analysis uses API frequencies along with ensemble based classifiers where in place of a single classifier the approach uses multiple classifiers. In the experiments performed malicious behaviors of 100 malware samples are analyzed and 24 APIs that are more frequently used in the malware are selected. A sandbox tool is used to measure the API's frequencies with APIs frequency defined as the ratio between the number of invocations of specific APIs and the total invocations of APIs. The authors perform analysis by applying these frequencies using ensemble based classifiers and in the end measure the accurateness of the implemented analysis.

Wagener et al. [8] proposed a malware detection method based on the API invocation information used to call system functions by the malware. In this approach binary code is defined for every API and this binary code is used to record malware patterns of API invocation. The authors then compared the API invocation information using an edit distance matrix and their own formulas and finally measured among the malware variants the prevalent similarities.

Xu et al. [9] devised a Windows systems based malware detection method using a custom built PE binary parser tool instead of third-party disassemblers as it extracts useless malware features degrading the analysis systems performance. This PE binary parser tool is used to analyze the PE binary code and generate API call sequences and in case they are known malware these sequences are recorded in signature database system. A comparison of these sequences with unidentified malware API call sequences and the similarities are calculated between the two types of sequences finally to find new malware.

Liu Wu et al. [10] devised Malware Behavior Feature (MBF) approach that uses Boolean expressions to extract malware features based on the invocation patterns of malicious API. A malware detection algorithm is devised based on the MBF approach and this approach is implemented with a malware detection system based on Malware Behavior Feature. The model achieves in tests results a high rate of accuracy and also detects malware that are not known before.

Martin Apel et al. [11] for identifying polymorphic malware experimented with various distance measures and their beneficial properties. The study evaluates and compares different distance metrics for detecting malware behavior features and uses a suitable distance measure to find behavior similarities that are used to group the malware samples.

Mamoun Alazab et al. [12] devised approach analyzes the usage of the features of API invocation for malicious intents. The method automatically extracts features of API invocation in binary executable files and performs automatic analysis of these features and their classified behaviors for unknown malicious intents in the maliciously bundled files. The authors use a four-step procedure based fully automatic system to group the features into six important classes in terms of their suspicious behavior features in the invocation of API.

The above discussed model uses features of API invocation for detecting malicious behavior. The constraint of this model is it gives more emphasis on the way to describe the malicious behavior and not the multifaceted approaches required to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

compare and produce results of similarities existing in the malwares with greater accuracy. In API features comparison it is highly challenging to apply complicated algorithms. In order to enhance the efficiency of a malware detection system, it is obvious to invite advanced techniques for calculating the similarities in malware variants.

In regard to surpass the issues elevated in malware attack detection, our contribution adapted dynamic analysis to diagnose the malware [13] [14]. The proposed malware attack detection strategy uses API call sequences to notify the behavioral act of malware, which is since the APIs are the core interfaces to perform operating system level malevolent activities. The exploratory scale that proposed here in this manuscript is hence considering the n-gram call sequences as features. Unlike the traditional strategy of sequence comparison to identify the malware, the proposed ESMP identifies the impact of the n-gram sequences based on the proposed scale, since the call sequence similarity comparison is approximate and often fails due to slight noise in call sequence. The call sequence similarity comparison prefers n-gram sequences to compare, in contrast to that our proposed scale prefers 2 gram sequences as features in order to withhold the impact of call sequence noise. Therefore, our proposed method can identify malevolent sequences under noise impact.

3. EXPLORATORY SCALE FOR MALWARE PERCEPTION

Our proposed ESMP devises exploratory scale to malevolent scope and exploratory scale to benevolent scope. In order to this, 2 gram sequences of the given malevolent and benevolent call sequences are extracted as respective features. Further optimal features respective to malevolent and benevolent call sequences given will be tracked, which is done by sequential floating forward search [15]. Then the impact of each feature and each call sequence of given malevolent and benevolent call sequences given as training set, which will be assessed using bipartite graph strategy. Then these respective impacts of features and call sequences will be used to define exploratory scale to malevolent scope and exploratory scale to benevolent scope.

The objective of the ESMP defines scale by the knowledge gained from the training set. In order to this the given training set tds is partitioned into two by their Boolean labels (true, false or 0, 1). The records of partition tlr labeled as true are the call sequence records confirmed as malicious and the records of partition flr labeled as false are call sequence records confirmed as benevolent. Then these two partitions tlr and flr are used to define "Exploratory Scale to Malicious Scope ($esms$)" and "Exploratory Scale to Benevolent Scope ($esbs$)". Further these scales are used to gauge the attack scope of a given call sequence record. The approach that used in both cases of defining $sdhs$ and $snhs$ are the similar that explained in following sections.

3.1 Exploring Call Sequence set

The dataset DS of call sequence records of size $|DS|$ will be considered for training towards defining the both exploratory scales $esms$ and $esbs$. Each unique two calls in sequence of given call sequence records are considered as features and each two calls in sequence referred as feature in further description.

The given call sequence set DS partitioned in to $MCS = \{mcs_1, mcs_2, ..., mcs_{|MCS|}\}$ and $BCS = \{bcs_1, bcs_2, ..., bcs_{|BCS|}\}$ that represents malicious and benevolent call sequences respectively. The feature sets tcs_M, tcs_B of respective datasets MCS and BCS contains all unique two calls in sequence found in respective call sequence records of datasets MCS and BCS as features, which are represented as follows

$tcs_M = \{m_1, m_2, ...m_{|tcs_M|}\}$ // two calls in sequence as features found in call sequence records of malicious call sequence set MCS

$tcs_B = \{b_1, b_2, ...b_{|tcs_B|}\}$ // two calls in sequence as features found in call sequence records of benevolent call sequence set BCS

3.2 Feature Optimization

The feature optimization is done by selecting features using one of the prominent feature search technique called sequential floating forward search [15]. This technique verifies the each feature optimality on a given criteria in a sequential pattern and also verifies the shoddiness of the feature under the given criteria and eliminates shoddier features. Such that this technique iterates by opting optimal feature or deleting shoddier features until

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

no change found in the optimal feature set. Since the optimal feature set size increases if new feature added and decreases if shoddier feature delete, the technique is referred as sequential floating forward search.

The context of our contribution verifies the optimality of feature such that a feature $\{m \exists m \in tcs_M\}$ of malicious call sequences should have high coverage (appears in majority of the given malicious call sequences) in MCS should evince least coverage (appears in least number of benevolent call sequences) in BCS. If feature contradict to this property then the feature said to shoddier feature towards malicious call sequences. Similarly, the optimal features of tcs_B will be found. The property that considered to select optimal features from tcs_B is, a feature $\{b \exists b \in tcs_B\}$ of malicious call sequences should have high coverage (appears in majority of the given benevolent call sequences) in BCS should evince least coverage (appears in least number of malicious call sequences) in MCS. If feature contradict to this property then the feature said to shoddier feature towards malicious call sequences.

The feature optimization process discards all shoddier features from the tcs_M and tcs_B respectively.

3.3 Defining Exploratory Scales from Training Data

The exploratory scale for respective malicious and benevolent call sequence detection is described in this section. In order to this, the correlation between each pair of features (pair of two calls in sequence), which is used further to assess the confidence of each feature (two call in sequence) towards each call sequence of the respective MCS and BCS. This feature confidence is used further to compute the confidence of each call sequence of respective MCS and BCS. The feature confidence and call sequence confidence obtained will be used further to identify the impact of each feature and call sequence respective to malicious and benevolent scope. These feature and call sequence impacts are then used to define the exploratory scales $esms$ and $esbs$ to detect malicious and benevolent call sequence respectively.

3.3.1 Measuring Feature (two calls in sequence) pair correlation

In order to define the scale, initially for each feature set tcs_M and tcs_B, the correlation between each pair of two calls in sequence will be assessed as follows:

$\forall_{i=1}^{|tcs_M|} \{m_i \exists m_i \in tcs_M\}$ Begin

$\forall_{j=1}^{|tcs_M|} \{m_j \exists m_j \in tcs_M\}$ Begin

$$\rho_{\{m_i, m_j\}} = \frac{\sum_{k=1}^{|MCS|} \{1 \exists (i \neq j) \wedge \{m_i, m_j\} \subseteq mcs_k\}}{|MCS|}$$

// the number of malicious call sequences contain both features divides by total number of malicious call sequences

End

End

Similarly, the feature pair correlation for each pair of two calls in sequence will be assessed as follows:

$\forall_{i=1}^{|tcs_B|} \{b_i \exists b_i \in tcs_B\}$ Begin

$\forall_{j=1}^{|tcs_B|} \{b_j \exists b_j \in tcs_B\}$ Begin

$$\rho_{\{b_i, b_j\}} = \frac{\sum_{k=1}^{|BCS|} \{1 \exists (i \neq j) \wedge \{b_i, b_j\} \subseteq bcs_k\}}{|BCS|}$$

// the number of benevolent call sequences contain both features divides by total number of benevolent call sequences

End

End

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.3.2 The Measuring feature to call sequence confidence

Each feature confidence towards respective each call sequence of MCS and BCS can be estimated as follows:

$\forall_{i=1}^{|MCS|} \{mcs_i \exists mcs_i \in MCS\}$ Begin

$\forall_{j=1}^{|tcs_M|} \{m_j \exists m_j \in tcs_M \wedge mcs_i \ni m_j\}$ Begin

$$\chi_{m_j \Rightarrow mcs_i} = \frac{\sum_{k=1}^{|tcs_M|} \left\{ \rho_{(m_j, m_k)} \exists (k \neq j) \wedge (m_k \in tcs_M) \wedge (mcs_i \ni m_k) \right\}}{|mcs_i| - 1}$$

// aggregation of correlation of each pair of features contain feature m_j divides by total number of features (two call in sequence) found in mcs_i.

End

End

Similarly, each feature confidence towards each call sequence of BCS is measured as follows:

$\forall_{i=1}^{|BCS|} \{bcs_i \exists bcs_i \in BCS\}$ Begin

$\forall_{j=1}^{|tcs_B|} \{b_j \exists b_j \in tcs_B \wedge bcs_i \ni b_j\}$ Begin

$$\chi_{b_j \Rightarrow bcs_i} = \frac{\sum_{k=1}^{|tcs_B|} \left\{ \rho_{(b_j, b_k)} \exists (k \neq j) \wedge (b_k \in tcs_B) \wedge (bcs_i \ni b_k) \right\}}{|bcs_i| - 1}$$

// aggregation of correlation of each pair of features contain feature b_j divides by total number of features (two call in sequence) found in bcs_i.

End.

End.

3.3.3 Measuring Call Sequence Confidence

The call sequence confidence of respective MCS and BCS is measured further as follows

$\forall_{i=1}^{|MCS|} \{mcs_i \exists mcs_i \in MCS\}$ Begin

$$\chi_{mcs_i} = \frac{\sum_{j=1}^{|tcs_M|} \left\{ \chi_{m_j \Rightarrow mcs_i} \exists mcs_i \ni m_j \right\}}{|mcs_i| - 1}$$

// aggregation of confidence of all features towards respective call sequence divides by total number of features (two call in sequence) found in mcs_i.

Similarly, the confidence of each call sequence of respective BCS is assessed as follows

$\forall_{i=1}^{|MCS|} \{bcs_i \exists bcs_i \in BCS\}$ Begin

$$\chi_{bcs_i} = \frac{\sum_{j=1}^{|tcs_B|} \left\{ \chi_{b_j \Rightarrow bcs_i} \exists bcs_i \ni b_j \right\}}{|bcs_i| - 1}$$

// aggregation of confidence of all features towards respective call sequence divides by total number of features (two call in sequence) found in bcs_i.

3.3.4 Measuring feature impact

The next level of scale definition, the feature impact respective to MCS and BCS are assessed as follows

$\forall_{i=1}^{|tcs_M|} \{m_i \exists m_i \in tcs_M\}$ Begin

$$\iota_{m_i} = \frac{\sum_{k=1}^{|MCS|} \left\{ \chi_{mcs_k} \exists mcs_k \ni m_i \right\}}{\sum_{l=1}^{|MCS|} \chi_{mcs_l}}$$

// aggregation of the confidence of call sequences those contains respective feature divides by the aggregation of confidence of all call sequences found in MCS

End

Similarly, the feature impacts of the respective BCS is measured as follows:

$\forall_{i=1}^{|tcs_B|} \{b_i \exists b_i \in tcs_B\}$ Begin

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$\iota_{b_i} = \frac{\sum_{k=1}^{|BCS|} \{\chi_{bcs_k} \exists bcs_k \ni b_i\}}{\sum_{l=1}^{|BCS|} \chi_{bcs_l}}$$

// aggregation of the confidence of call sequences those contains respective feature divides by the aggregation of confidence of all call sequences found in BCS

End

3.3.5 Measuring Call Sequence Impact

Assessing respective call sequence impact towards malicious or benevolent scope is the next in hierarchy of exploratory scale definition. The call sequence impact of each call sequence in MCS is assessed as follows

$\forall_{i=1}^{|MCS|} \{mcs_i \exists mcs_i \in MCS\}$ Begin

$$\iota_{mcs_i} = \frac{\sum_{j=1}^{|tcs_M|} \{\iota_{m_j} \exists mcs_i \ni m_j\}}{\sum_{k=1}^{|tcs_M|} \{\iota_{m_k}\}}$$

// aggregation of the impact of the all features exists in respective call sequence divides by the aggregation of impact of all features found in tcs_M

End

Similarly, the respective call sequence impact of all call sequence of BCS can be found as follows

$\forall_{i=1}^{|BCS|} \{bcs_i \exists bcs_i \in BCS\}$ Begin

$$\iota_{bcs_i} = \frac{\sum_{j=1}^{|tcs_B|} \{\iota_{b_j} \exists bcs_i \ni b_j\}}{\sum_{k=1}^{|tcs_B|} \{\iota_{b_k}\}}$$

// aggregation of the impact of the all features exists in respective call sequence divides by the aggregation of impact of all features found in tcs_B

End

3.3.6 Exploratory Scale to Malicious and Benevolent Scope

Further the impact of the respective call sequences of MCS and BCS are used further to define exploratory scales $esms$ and $esbs$ respectively, which is as follows:

$$esms = \frac{\sum_{i=1}^{|MCS|} \{\iota_{mcs_i} \exists mcs_i \in MCS\}}{|MCS|}$$

Further the lower bound and upper bound of the $esms$ is estimated by differentiating the mean absolute distance [17] of the impact of all call sequence in MCS, which is as follows:

$$esms_{mad} = \frac{\sum_{i=1}^{|MCS|} \sqrt{(esms - \iota_{mcs_i})^2}}{|MCS|}$$

// Mean absolute distance of $esms$

$esms_l = esms - esms_{mad}$ // lower bound of the $esms$

$esms_u = esms + esms_{mad}$ // upper bound of the $esms$

Similarly, the exploratory scale to benevolent scope ($esbs$) and its lower and upper bounds for respective call sequences of BCS can be measured as follows:

$$esbs = \frac{\sum_{i=1}^{|BCS|} \{\iota_{bcs_i} \exists bcs_i \in BCS\}}{|BCS|}$$

$$esbs_{mad} = \frac{\sum_{i=1}^{|BCS|} \sqrt{(esbs - \iota_{bcs_i})^2}}{|BCS|}$$

// Mean absolute distance of $esbs$

$esbs_l = esbs - esbs_{mad}$ // lower bound of the $esbs$

$esbs_u = esbs + esbs_{mad}$ // upper bound of the $esbs$

3.4 Scaling Call sequence Record

The Exploratory Scale to Malicious Scope $(esms)$, Exploratory Scale to Benevolent

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Scope $(esbs)$ and their respective lower and upper bounds devised (see section 3.4) will be used further to assess the state reflected by a given call sequence record as follows

Let cs be the given call sequence, the impact of cs towards malicious scope can be assessed as follows.

$$\iota_{cs \Rightarrow MCS} = \frac{\sum_{i=1}^{|tcs_M|} \left\{ \iota_{m_i} \exists m_i \in tcs_M \wedge cs \ni m_i \right\}}{\sum_{j=1}^{|tcs_M|} \left\{ m_j \exists m_j \right\}}$$

// the aggregate of each feature impact of that exists in tcs_M and cs divides by the aggregate of all features exists in tcs_M

Further the impact of cs towards benevolent scope is assessed as follows:

$$\iota_{cs \Rightarrow BCS} = \frac{\sum_{i=1}^{|tcs_B|} \left\{ \iota_{b_i} \exists b_i \in tcs_B \wedge cs \ni b_i \right\}}{\sum_{j=1}^{|tcs_B|} \left\{ \iota_{b_j} \exists b_j \in tcs_B \right\}}$$

// the aggregate of each feature impact of that exists in tcs_B and cs divides by the aggregate of all features exists in tcs_B

//The average of influence weights of all features towards normal scope, which are belongs to given record $p\,r$

Then these malicious impact scope $\iota_{cs \Rightarrow MCS}$ and benevolent impact scope $\iota_{cs \Rightarrow BCS}$ of cs are used to estimate the given call sequence cs is attack prone or normal, which is as follows.

$\left(\iota_{cs \Rightarrow MCS} \geq esms_u\right) \vee$

$\left(\iota_{cs \Rightarrow MCS} \geq esms \wedge \iota_{cs \Rightarrow BCS} \leq esbs\right)$

//Call sequence cs is scaled as malicious

$\left(\iota_{cs \Rightarrow BCS} \leq esbs_l\right)$

//Call sequence cs is scaled to attack prone (possibly zero day attack)

$\left(\iota_{cs \Rightarrow BCS} \geq esbs_u\right) \vee$

$\left(\iota_{cs \Rightarrow BCS} \geq esbs \wedge \iota_{cs \Rightarrow MCS} \leq esms_l\right)$

//Call sequence cs to be benevolent

Rest of all cases said to be suspicious and it is advised to recommend safe zone access

4 EXPERIMENTAL STUDY

The experimental study was done on the call sequence dataset called CSDMC2010_API [16]. This dataset contains 388 call sequences those labeled as 1 (malicious call sequence) and 0 (benevolent call sequence). In order to estimate the explorative scale, 75% of malicious and benevolent call sequences of the chosen dataset were used. The rest 25% call sequences were unlabeled and used to test the significance of the proposal towards malware detection accuracy. The results obtained from experimental study were listed in table 1, which are evincing the significance of the ESMP towards malware detection. The detailed prediction statistics were explored in table 2.

Table 1: The Traininginputs And Resultant Exploratory Scales

Malicious Call Sequence	240
Benevolent Call Sequence	51
$esms$	0.574996
$esms_{mad}$	0.10572
$esms_l$	0.475628
$esms_u$	0.683246
$esbs$	0.644838
$esbs_{mad}$	0.179485
$esbs_l$	0.47749
$esbs_u$	0.81056

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: The Testing Inputs And Prediction Statistics Of The ESMP

Malicious Call Sequence	80
Benevolent Call Sequence	17
True Positives	79
True Negatives	15
False Positives	2
False Negative	1
positive predictive value	0.975308642
Negative Predictive Value	0.9375
Accuracy	0.969072165
Sensitivity	0.9875
Specificity	0.882352941

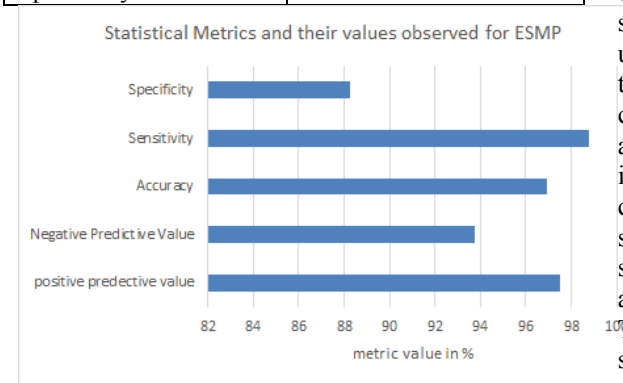

Figure 1: Visualization Of Prediction Metrics And Their Values Observed For ESMP

The 97 call sequences were used to assess the significance of the proposed ESMP. Among them, 80 call sequences are malicious and 17 are benevolent call sequences. The ESMP assessed the given input call sequences such that 79 call sequences are true positives (true prediction of malicious call sequences), 2 call sequences are false positive (false prediction of benevolent call sequences as malicious), 15 call sequences are true negatives (true prediction of benevolent call sequences) and 1 call sequences is false negative (malicious call sequence predicted as benevolent). Hence the malicious call sequence prediction value (also known as precision) is 0.975, benevolent prediction value is 0.938, the malicious call sequence detection rate (also known as sensitivity) is 0.988, the benevolent call sequence detection rate (also known as specificity) is 0.882 and the accuracy (which is the ratio between true prediction of malicious and benevolent call sequences and all given call sequences) is 0.97. These statistics indicating that the ESMP is significant to differentiate the malicious and benevolent call sequences with 97% accuracy. The sensitivity of the ESMP is 99%, hence the zero day malicious call sequences prediction is at its best. The prediction statistics observed from the experimental study of the ESMP are visualized in fig1.

5 CONCLUSION:

An Exploratory Scale for Malware Perception (ESMP), which is based on heuristics learned from API call sequence was proposed in this manuscript. The proposed model delivers an exploratory scale to estimate a call sequence is malicious or benevolent. The model proposed is considering the 2 gram call sequences as features of the given training set of malicious and benevolent call sequences. Further, selects optimal features by using sequential floating forward search [15]. Then the impact of these optimal features and respective call sequences towards the scope of maliciousness and benevolence were assessed. Afterwards these impact ratios observed for respective features and call sequences were used to devise exploratory scale of malicious act scope and benevolent act scope. These scales further can be used to percept a given call sequence is malicious or benevolent. The prediction phase extracts all 2 gram call sequences from the given input call sequence and estimates the impact of these features. Further uses these impacts of the respective features to estimate the impact of the given call sequence towards the scope of maliciousness and the scope of benevolence. Then these values are compared to exploratory scales devised for malicious scope and benevolent scope to percept the given call sequence is malevolent or benevolent. The experimental study evinces the significance (malevolent detection rate is 99%) and accuracy (malevolent and benevolent call sequence detection rate is 97%) of the ESMP to identify a call sequence is malevolent or benevolent. This work can be extended to identify the impact of 3 gram and n gram call sequences ad features and also considering the correlation of 2, 3 and n gram call sequences correlation that minimizes the learning overhead in exploratory scale definition. The other future research direction would define evolutionary computation or soft computing strategies those uses the proposed exploratory scale as cost or objective function.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES

[1] M. Egele, T. Scholte, E. Kirda, and C. Kruegel. A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys (CSUR), 44(2):6, February 2012.

[2] Y. Qiao, Y. Yang, L. Ji, and J. He. Analyzing malware by abstracting the frequent itemsets in api call sequences. In Proc. of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'13), Melbourne, VIC, pages 265–270. IEEE, July 2013.

[3] B. Kang, T. Kim, H. Kwon, Y. Choi, and E. G. Im. Malware classification method via binary content comparison. In Proc. of the 1st 2012 ACM Research in Applied Computation Symposium (RACS'12), New York, USA, pages 316–321. ACM, October 2012.

[4] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala. Static analyzer of vicious executables (save). In Proc. of the 20th Computer Security Applications Conference (ACSAC'04), Tucson, AZ, USA, pages 326–334. IEEE, December 2004.

[5] C. Willems, T. Holz, and F. Freiling. Toward automated dynamic malware analysis using cwsandbox. ACM Computing Surveys (CSUR), 5(2):32–39, March 2007.

[6] G. Hunt and D. Brubacher. Detours: Binary interception of win32 functions. In Proc. of 3rd USENIX Windows NT Symposium (WINSYM'99), Seattle, WA, USA, pages 135–144. USENIX Association, July 1999.

[7] P. Natani and D. Vidyarthi. Malware detection using api function frequency with ensemble based classifier. In Proc. of the 1st Security in Computing and Communications (SSCC'13), Mysore, India, LNCS, volume 377, pages 378–388. Springer-Verlag, August 2013.

[8] G. Wagener, A. Dulaunoy, et al. Malware behaviour analysis. Journal in computer virology, 4(4):279–287, November 2008.

[9] J.-Y. Xu, A. H. Sung, P. Chavez, and S. Mukkamala. Polymorphic malicious executable scanner by api sequence analysis. In Proc. of the 4th International Conference on Hybrid Intelligent Systems (HIS'04), Kitakyushu, Japan, pages 378–383. IEEE, December 2004.

[10] L. Wu, R. Ping, L. Ke, and D. Hai-xin. Behavior-based malware analysis and detection. In Proc. of the 1st International Workshop on Complexity and Data Mining (IWCDM'11), Nanjing, Jiangsu, pages 39–42. IEEE, September 2011.

[11] M. Apel, C. Bockermann, and M. Meier. Measuring similarity of malware behavior. In Proc. of the IEEE 34th Conference on Local Computer Networks (LCN'09), Zurich, Swiss, pages 891–898. IEEE, October 2009.

[12] M. Alazab, S. Venkataraman, and P. Watters. Towards understanding malware behaviour by the extraction of api calls. In Proc. of the 2nd Cybercrime and Trustworthy Computing Workshop (CTC'10), Ballarat, VIC, pages 52–59. IEEE, July 2010.

[13] K.-S. Han, I.-K. Kim, and E. G. Im. Malware classification methods using api sequence characteristics. In Proc. of the 1st International Conference on IT Convergence and Security (ICITCS'12), Suwon, Korea, LNCS, volume 120, pages 613–626. Springer-Verlag, December 2012.

[14] K.-S. Han, I.-K. Kim, and E. G. Im. Detection methods for malware variant using api call related graphs. In Proc. of the 1st International Conference on IT Convergence and Security (ICITCS'12), Suwon, Korea, LNCS, volume 120, pages 607–611. Springer-Verlag, December 2012.

[15] Somol, P., Pudil, P., Novovičová, J., & Paclık, P. (1999). Adaptive floating search methods in feature selection. Pattern recognition letters, 20(11), 1157-1163.

[16] http://csmining.org/index.php/malicious-software-datasets-.html (downloaded on May 15, 2016)

[17] Carmines, E. G. (1979). Reliability and validity assessment. Sage publications.