International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 8 Issue 1 –APRIL 2014.

ARTIFICIAL BEE COLONY ALGORITHM USING DATASET FEATURE FILTERING FOR MALWARE DETECTION

R.Ramya,
Department of CSE,
Chettinad College of Engg and Tech, Karur.

Ms.C.Sangeetha,
Assistant Professor,
Department of CSE,
Chettinad College of Engg and Tech, Karur.

Dr.T.Rajendran
Professor and Head,
Department of CSE & IT,
Chettinad College of Engg and Tech, Karur.

***Abstract*—N-gram analysis is an approach that investigates the structure of a program using bytes, characters, or text strings. A key issue with N-gram analysis is feature selection amidst the explosion of features that occurs when N is increased. The experiments within this paper represent programs as operational code (opcode) density histograms gained through dynamic analysis. The analysis of opcode density features using supervised learning machines performed on features obtained from run-time traces. A support vector machine is used to create a reference model. Proposed work we intend to expand the detection methods by investigating N-gram size, which will dramatically increase the number of features. . A novel approach for feature selection is introduced using CHABCF, (Chaotic Artificial Bee Colony), algorithm. Combination of paradigms: (1) Chaos theory (2) Artificial Bee Colony optimization. The system is used for ambiguity removal while chaos is used for generating the initial population of our bee colony optimization algorithm.**

IndexTermsKNN,metamorphismmalware,obfuscation,packers,polymorphism, SVM, Bee Colony

1. INTRODUCTION

Latest years have glimpsed massive development in malware, with signature detection and supervising supposed cipher for renowned security vulnerabilities evolving ineffective and troublesome. In answer, investigators need to take up new detection advances that outmanoeuvre the distinct strike vectors and obfuscation procedures employed by them alware advances to filtering out irrelevant features and writers. Detection advances that use the host environment's native opcode sat run-time will circumvent numerous of the malware writers' endeavours to avoid detection. One such approach, as suggested in this paper, is the analysis of opcode density characteristics utilizing overseen learning appliances performed on characteristics got from run-time traces. In future study we propose to elaborate the detection methods by enquiring N-gram size, which N will spectacularly boost the number of characteristics. With this anticipated blast of characteristics we have chosen to enquire procedures to prune irrelevant features. While standard constituent investigation (PCA) is a well-liked procedure to decrease features in subspace, this paper aspires to recognise characteristic decrease in the original dataset space.

For large datasets, or exorbitant (computation) expanse purposes, the teaching process affiliated with discovering appliances can become immense. Thus, the characteristic explosion that happens with N-grams for large standards of N needs to be addressed. This paper investigates three starts, in Section II, with a consideration on associated research. In part III, the trials are placed into context with an overview of the untested approach. partIV specifies the natural environment used to capture the dataset and inserts anti-analysis approaches taken by malware writers. This is pursued, in part V, with an interpretation of how the dataset is conceived. The Support Vector appliance (SVM) is introduced in part VI and describes the creation of a reference model that is utilised to validate the successfulness of the subsequent filter trials. Section VII investigates three filters: Firstly, a simple hypotheses test is considered to determine the prospect that the benign and malicious dataset do not pertains to the identical circulation; secondly, an in-depth look at the circulation by assessing the locality of intersect between the benign and malicious distributions; and eventually, a gaze at the projection of the dataset into a subspace utilising eigenvalues. Section VIII summarizes the outcomes and key characteristics recorded during these trials. Finally, part IX concludes the paper by comparing the results with other study and minutia future work that will be conveyed outas part of this research.

2. RELATED WORK

Comprehensive study has been undertaken into the detection of malicious code utilising both static and dynamic analysis. Malware research can be categorized, not only in periods of static and dynamic investigation, but furthermore in how the information is processed after it is captured. well liked research procedures include: command Flow Graphs (CFG) for both course and fine kernel investigation, state appliances to form scheme demeanour, the mapping of stack procedures and N-gram investigation.

Lakhotia et al. presented a state appliance method to detect obfuscated calls pertaining to impel pop and ret opcodes that are mapped to stack procedures. although, their approach did not form positions where the push and burst instructions are decomposed into multiple instructions, such as exactly manipulating the stack pointer using mov instructions. Bilar utilised static investigation to get opcode distributions from PE documents that could be utilized to recognise polymorphic and metaphoric malware. Bilar’s outcome show that numerous common opcodes (mov, push, call, etc.) did not make good signs of malware. although, lesser frequent opcodes such ja, adc, sub, inc and add proved to be better indicators of malware.

Santos et al. analyzed the likeness between families of malware and the dissimilarity between malware and benign software utilizing opcode-sequence profited through static analysis of PE documents. Santos outcome displayed that utilising (N-gram) weighted opcode frequency a high degree of likeness existed between families of malware, but the likeness ranking between malicious and benign programs was too high to be an productive classifier.

However, using made a larger distinction between malicious and benign software. In a subsequent paper, Santos examined an SVM with a single-class discovering approach that utilised the frequency of opcodes got from static investigation. Santos reduced the effort of labelling that is needed for the training stage and highlights the topic of unpacking malware. Santos et al. assessed some learning procedures (KNN, Bayesian mesh, SVM, etc.) and displayed that malware can be noticed with a high degree of correctness using opcode-sequence.

In this vein, we have chosen to focus our research on the identification of malware utilizing opcodes. However, we have selected to get the opcodes from run-time program traces. Bilar illustrated that

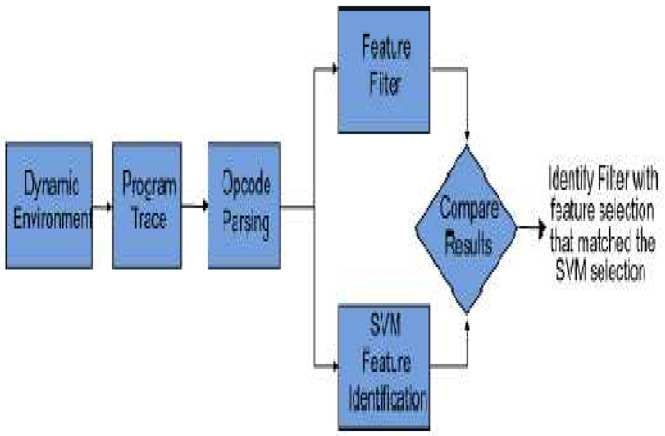
Feature Filter
Dynamic Environment
Program Trace
Opcode Parsing
Compare Results
Identify Filter with feature selection that matched the SVM selection
SVM Feature Identification

Figure.1. Experiment overview

opcode N-gram could be utilised to notice malware.thus, we started with and have demonstrated that malware can be recognised with a decreased set of features. primary enquiries display that for , 149 characteristics are made in the raw dataset made 8092 characteristics (no filtering) thus, before extending the study by expanding N, it is prudent to set up a cornerstone on which to filter the dataset to avert characteristic explosion. To this end, this research focuses on finding a filter to remove redundant features.

3. SYSTEM OVERVIEW

The motivation for this study is to decrease the computational overhead needed when N-gram investigation is performed on low-level fine kernel data. Thus, developing a lightweight filter that will reduce the number of characteristics to be processed will in turn decrease the computational overhead; therefore making the training stage of the SVM approach a viable answer for N-gram analysis where large feature groups are generated. Fig. 1 shows an overview of the untested approach taken in this paper. The programs under enquiry are run in a check natural environment with a debug tool supervising the runtime opcodes. After culmination, the data is parsed into opcode histograms and after some conditioning the dataset is passed to the SVM to assemble a quotation model. The quotation form is constructed by configuring the SVM to present an exhaustive seek by crossing through all the characteristics, seeking for those opcodes that have a positive influence on the classification of benign and malicious programs. To assess the diverse filtering algorithms, each filter processes the initial dataset in an attempt to duplicate the identical reference model made by the SVM.

4. DATASET CREATION

Operational ciphers (Opcodes) are appliance dialect directions that perform CPU operations on operands such as arithmetic, memory/data manipulation, ordered operations and program flow command. There are 15 opcodes exactly mentioned to in this paper, which are grouped as follows: 1) Arithmetic operations—add , adc (add with convey flag), inc, sub; 2) recollection manipulation—lea (load productive address), mov, burst, impel (retrieve and location facts and figures up on a stack); 3) Logical operations—xor (exclusive OR); 4) Program flow control—call (jump to a function), ret (return from function) cmp (compare data), ja, je (jump if a condition is met); rep (a prefix that does again the particular operation). The dataset is assembled by representing each executable document as a set of opcode density histograms obtained from runtime traces. Note that the operands affiliated with each opcode are omitted and that only the opcodes are recorded. Classification jobs engage dividing facts and figures into teaching and test facts and figures. Each training-set example is allotted a goal value/label i.e., benign or malicious. The aim of the SVM is to assemble a form that predicts the goal standards of the test facts and figures. There are 260 benign Windows XP documents taken from the 'Program documents' book or directions (training documents 230, validation documents 30). There are 350 malware documents (training documents 310, validation documents 40) which are malicious windows executable documents downloaded from available) and consists of a range of malicious undertakingssuchas:backdoor,downloaders,schemestrike,forgeryalert/warnings,Ad-Aware,datastealer.

To ensure that Ollydbg rightly unpacked and ran the malware, trials were constrained to programs that ollydbg rightly recognised as crammed or encrypted. The malware samples were run for 3 minutes ensuring that not only the loading and unpacking stages were noted but further more that malicious undertaking appeared, i.e., pop-up, composing to the disk or registry files. While there are 344 Intel opcodes, only 149 distinct opcodes are noted throughout the apprehended datasets for all programs traced throughout this trial. The dataset is normalized by assessing the percentage density of opcodes rather than the unconditional opcode count to eliminate time variance introduced by distinct run extents of the various programs. The dataset is arranged into most routinely happening opcodes.

An primary evaluation of the data displays two key properties a) The circulation of the various opcodes does not conform to any reliable distribution form; rather opcode distribution varies substantially as showed by the distinction between the mov and ret opcodes, recounted later in VII: 'Area of Intersect'. Therefore, no one data form could be presumed and hence a nonparametric procedure should be utilised. b) The facts and figures standards are a percentage of the opcodes inside a particular program. For demonstration, 0 means that the opcode does not occur inside that program find or 0.25 means that 25% of the program find comprises of that opcode. To advance the performance of the SVM the facts and figures is linearly leveled (0,+1).

5. SUPPORT VECTOR MACHINE

Support Vector appliance (SVM) is a technique utilised for facts and figures classification and was presented by Boser et al. in 1992 and is categorized as a kernel procedure. The kernel procedure algorithm depends on dot-products function, which can be replaced by other kernel functions that map the facts and figures into a higher dimensional feature space. This has two benefits: foremost, the ability to develop a nonlinear decision plane and secondly, permits the user to request a classification to data that does not have an intuitive approach i.e., SVM training when the facts and figures has a non regular or unidentified circulation.

The facts and figures set comprises of 149 different opcodes, each having their own exclusive distribution characteristics and therefore a SVM is an befitting alternative. As cited earlier, the facts and figures is linearly levelled to advance the performance of the SVM. The major advantages of scaling are a) it avoids attributes with larger numeric varieties overriding those with lesser numeric varieties and b) it avoids numerical adversities throughout the calculation as kernel values usually count on the inner products of characteristic vectors, for example, in the case of the linear kernel and the polynomial kernel, large ascribe values might origin numerical difficulties.

SVM is used to conceive a quotation form to validate the filter trials that are presented in the subsequence parts. The SVM is configured to cross through the dataset seeking for opcodes that have a affirmative influence on the classification of benign and malicious programs. The search begins with six opcodes scanning over the complete facts and figures sequence for all exclusive permutations for that number of opcodes. The seek is recurring for five opcodes and then four opcodes.

Keypointstonoteare:

1) The 6 opcodes ja, adc, sub, inc, add and rep, each having an significance ranking of more than 20% of the top detection rate, are chosen as the most important indicators for classifying benign and maliciousprograms.

2) mov has a negative influence on the classification and identification of software. i.e., when mov is part of the analysis facts and figures the output/classification is always incorrect. The mov has a high density (30% and 40% in the presented dataset) in both benign and malicious programs.

6. PROPOSED OPCODEPREFILTERING APPROACH

N-gram analysis presents a dimensionality difficulty in terms of the number of raw characteristics produced and if left unfiltered would outcome in a high computation cost during the SVM teaching stage. To reduce this effort and slender the locality of search, this study aims to recognise filters that can choose the optimum features former to feeding them to a SVM.

This part investigates two advances to filtering irrelevant opcodes. Starting with an enquiry into the ’area of intersect’ between benign and malicious distributions using Linear programming methods and then concludes with an enquiry into subspace investigation utilising Principle constituent investigation (PCA) and Eigenvectors.

A.Area of Intersect

Secondly, consider the simplistic characteristics of benign and malicious percentage of a given program that is made up of a specific opcodes with a usual circulation as shown in Fig. 4. The plots are grouped into density bends for benign and malicious software of a lone opcode. The level axis relates to the opcode and the upright axis shows the number of programs with that percentage of opcode. The key characteristic to note is the overlapping area of the two density bends. The larger the distinction between the signify of the curves and narrower the standard deviation decreases the overlapping locality and thus decreases the interference and corresponding misclassification of the benign and malicious software.

Constraints—The data is in the pattern of a probability density curve. The level axis comprises the makeup of a program i.e., the opcode percentage that makes up a program and the upright axis, comprising the number of programs that have that percentage of opcodes. The likelihood density is founded on a percentage of opcode counts obtained from traces throughout the execution of a program. The minimum worth is 0 and the greatest is the percentage of the most happening opcode within the apprehended dataset (mov). therefore the maximum value is 0.4 (40%).

Conclusion Variable—this is the worth discovered throughout the search for the greatest or minimum issue. It is the percentage of a particular opcode that yields the utmost locality of benign and malicious density that lies either side of the decision plane. **Target function**—is the numerical sign used to characterize the aim of the task. The mathematical input to the LP is the cumulative likelihood as the conclusion variable is incremented across the variety. thus the greatest classification would be accomplished when the two density bends do not intersect and their whole locality lies on their respective edge of the conclusion plane.

B. Subspace

An alternative approach to work out the significance of the one-by-one opcodes, thereby grading their usefulness as classification characteristics, is to investigate the eigenvalues and eigenvectors in subspace. Principal constituent Analysis (PCA) is a transformation of the covariance matrix and it is defined as [21]:

$$C_{ij} = \frac{1}{n-1}\sum_{m=1}^{n}(X_{jm} - \overline{X}_i)\,(X_{jm} - \overline{X}_j).$$

Where

C Covariance matrix of PCA transformation;
X dataset value;
$\overline{X}$ dataset mean;
n and m data length;

This is a method utiliused to compress facts and figures by mapping the facts and figures into a subspace while retaining most of the data/variation in the data. It reduces the dimensionality by mapping the data into a subspace and finding a new set of variables (fewer variables) that represent the original facts and figures. These new variables are called primary constituents (PCs).

As PCA is an algorithm that operates on variance of facts and figures i.e., a covariance matrix of the teaching facts and figures set, which is calculated in Matlab as follows:

```
C = con(trainingData)
[V,λ] = eig(C)
d =diag.
```

assessing the important standards by multiplying the important eigenvector Column by the respective eigenvalues and then summing each row

$$R_k = \sum_{k=1}^{8} V \,.\, d_k$$

Where

R Sum of the matrix variance;
C Covariance
V eigenvector
λ EigenValue matrix;
d EigenValue scalar;

Given that the six SVM selected opcodes have been grouped into the peak twelve opcodes i.e., peak 8% thereby eliminating the 92% irrelevant opcodes makes this an productive filtering means to decrease features former to the SVM training phase.

7. FEATURE SELECTION BASED METHOD FOR MALWARE DETECTION

Novel approach is introduced for feature selection based on the popular artificial bee colony algorithm. A better diversity in the population has been obtained using chaos theory with has been employed for ranking bees in the bee colony algorithm. Chaos theory has been incorporated for three main purposes: (1) increasing diversity in the initial population, (2) finding the neighbourhood a food source and (3) generating random numbers. Firstly, the initial population is generated for each source using the features of dataset with the around aim of selecting the best possible ones. For this purpose, a random number should be produced between 1 and the total number of features in dataset. This number specifies the number of features which will be used after feature selection.

CONCLUSION

This paper, suggests the use of SVM as a means of identifying malware. It shows that malware, that is packed/encrypted, can be noticed utilising SVMs and by utilising the opcodes chosen by the SVM as a standard, recognised a prefilter stage using eigenvectors that can decrease the characteristic set and thus decrease the teaching effort. Foremost, the identification of a high population opcode: mov that is not only is a poor sign of benign/malicious programs, but inhibits the ability to rightly classify programs when utilised with other opcodes such as ja, adc, sub, inc, add and rep. Secondly, a subset of opcodes can be utilised to notice malware. although, the SVM analysis demonstrates that ja, adc and sub are powerful signs of malware as they are four times more expected to be used in the correct classification of malware than the next most important opcodes (inc). Enhanced work proposed a novel approach for feature selection by artificial bee colony algorithm. A chaotic function has been used in this method for generating a population with more diversity, and the fuzzy logic is used for fitness based ranking of each food source and also allocating bees to these sources.

ACKNOWLEDGMENT

The authors thank anonymous reviewers for their valuable comments. The research was supported in part by projects.

REFERENCES

[1] A. Lakhotia, E.U. Kumar, and M. Venable, “A procedure for noticing obfuscated calls in malicious binaries,” IEEE Trans. Software Eng., vol. 31, no. 11, pp. 955–968, Nov. 2005.

[2] D. Bilar, “Opcodes as predictor for malware,” Int. J. Electron. Security Digital Forensics, vol. 1, no. 2, pp. 156–168, 2007.

[3] D. Bilar, “Callgraph properties of executables and generative mechanisms,” AI Commun., exceptional topic on Network Anal. in Natural Sci. and Eng., vol. 20, no. 4, pp. 231–243, 2007

[4] I. Santos, Y. K. Penya, J. Devesa, and P. G. Garcia, “N-grams-based document signatures for malware detection,” S3Lab, Deusto Technological Found., 2009.

[5] R. Sekar, M. Bendre, D. Bollineni, and Bollineni, R. Needham and M. Abadi, Eds., “A fast automaton-based procedure for noticing anomalous program behaviors,” in Proc. 2001 IEEE Symp.Security and Privacy, IEEE Comput. Soc., Los Alamitos, CA, USA, 2001, pp. 144–155

[6] W. L. K. Wang, S. Stolfo, and B. Herzog, “Fileprints: Identifying document types by n-gram analysis,” in Proc. 6th IEEE announce. promise Workshop, Jun. 2005, pp. : 64–71.

[7] I. Santos, F. Brezo, J. Nieves, Y. K. Penya, B. Sanz, C. Laorden, and P. G. Bringas, “Opcode-sequence-based malwaredetection,” in Proc.2nd Int. Symp. Eng. Secure programs and Syst.(ESSoS),Pisa, Italy, Feb. 3–4, 2010, vol. LNCS 5965, pp. 35–43.
[8] I. Santos, F. Brezo, B. Sanz, C. Laorden, and Y. P. G. Bringas, “Using opcode sequences in single-class learning to notice unknown malware,” IET announce. Security, vol. 5, no. 4, pp. 220–227, 2011.

[9] I. Santos, F. Brezo, X. Ugarte-Pedrero, and Y. P. G. Bringas, “Opcode sequences as representation of executables for data-mining-based unidentified malware detection,” Inform. Sci., 2011.

[10] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, “Detecting unidentified malicious cipher by applying classification techniques on opcode patterns,” Security Informatics, vol. 1, pp. 1–22, 2012.

[11] R. Moskovitch, C. Feher, N. Tzachar, E. Berger,M.Gitelman, S.Dolev, and Y. Elovici, “Unknown malcode detection using opcode representation,”in *Proc. 1st Eur Conf. Intell. and Security Informatics (EuroISI08),* 2008, pp. 204–215.

[12] Y. Song, M. Locasto, and A. Stavro, “On the infeasibility of modeling polymorphic shellcode,” in *Proc. ACM Conf. Computer and Commun.Security*, 2007, pp. 541–551.

[13] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X.Wang, “Effective and efficient malware detection at the end host,” in *Proc. 18th Usenix Security Symp.*, 2009, pp. 351–366.

[14] P. Ferrie, The ultimate anti debugge reference May2011.

[Online].Available:http://pferrie.host22.com/papers/antidebug.pdf.

[15] X. Chen, "Towards an understanding of anti-virtualization and antidebuggingbehavior in modern malware," *ICDSN Proc.*, pp. 177–186, 2008.

[16] Philip O'Kane, SakirSezer, Kieran McLaughlin, and EulGyuIm, "SVM Training Phase Reduction Using Dataset Feature Filtering for Malware Detection" IEEE Trans. Security., March 2013.

[17] B. E. Bernhard, G. M. Isabelle, and V. N. Vladimir, H. Haussler, Ed., "A training algorithm for optimal margin classifiers," in *Proc. 5th Ann.ACM Workshop on COLT ACM Press*, Pittsburgh, PA, USA, 1992, pp. 144–152.

[18] C. Ko, M. Ruschitzka, and K. Levitt, "Execution monitoring of security- critical programs in distributed systems: A specification-based approach," in *Proc. 1997 IEEE Symp. Security and Privacy*, Oakland, CA, USA, May 1997, p. 175-1 87.

[19] C.-W. Hsu, C.-C.Chang, and C.-J. Lin, A Practical Guide to Support Vector Classification, Department of Computer Science National Taiwan University, Taipei, Taiwan, Apr. 15, 2010 [Online]. Available: http://www.csie.ntu.edu.tw/.

[20] R. Vanderbei, *Linear Programming: Foundations and Extensions Pub*. New York, NY, USA: Springer, 2000, ISBN: 0792373421.