Analysis of Machine Learning Techniques used in Malware Classification in Cloud Computing Environment

Ajeet Kumar
Maharaja Surajmal Institute of Technology, GGSIPU, New Delhi, India

Naman Sharma
Maharaja Surajmal Institute of Technology, GGSIPU, New Delhi, India

Abhishek Khanna
Maharaja Surajmal Institute of Technology, GGSIPU, New Delhi, India

Saurav Gandhi
Maharaja Surajmal Institute of Technology, GGSIPU, New Delhi, India

ABSTRACT
Study the behavior of malicious software, understand the security challenges, detect the malware behavior automatically using dynamic approach. Study various classification techniques and to group these malwares and able to cluster different malware into unknown group whose characteristics are not known. The classifiers used in this research are k-Nearest Neighbors (kNN), J48 Decision Tree, and n-grams. Based on the analysis of the tests and experimental results of all the 3 classifiers, the overall best performance was achieved by J48 decision tree with a recall of 96.3%.

General Terms
Machine Learning techniques, malware classification, cloud computing, pattern recognition.

Keywords
Malware, Opcode n-grams, Bytecode n-grams, malware behaviors; malware classification.

1. INTRODUCTION
Distinguishing the region of malignant code on a given host is a vital section of any watchman segment. The manual heuristic review of static malware investigation is no more thought to be successful and effective looked at against the high spreading rate of malware. Different confusion and polymorphism innovation let the conventional static mark based discovery techniques get to be wasteful and deficient. The new idea of cloud security requires fast and mechanized identification and characterization of noxious programming. Malwares have a place with the same family frequently had the same conduct designs in light of the fact that they regularly have the comparable reason and capacity. Malware is normally arranged [1] as per its spread technique and objective.

In this paper, the authors add to a strategy for grouping malware in light of malware conduct report. In synopsis, this paper makes commitments as tails: The authors depict the behavioral profile through the behavioral clues of malware recorded in virtual circumstances. The authors create the follow report, in .csv group which comprise all the document of. .Asm and .Byte augmentation with closest malware definition.

The authors remove the distinctive components by behavior unit string from this report and assembling various components into gatherings in perspective of the semantic association, then used these informed bundles as segments and change take after report into a high-dimensional vector space. The authors use upheld vector machine to pick up from a readiness set and after that the authors use the predefined model for test set. Similar investigation demonstrates that their system has higher precision and productivity.

2. RELATED WORK
Related work takes a shot at malware grouping are centered around two perspectives most near their work. One viewpoint is finding exact system for getting malware conduct report and the other is create suitable conduct representation of malware family as the info of machine learning systems. Monitoring API call historial was used to discover program behavior [2], malicious program often run in a protective environment called ‘sandbox’ such as **CWSandbox** [3] and Anubis [4]. But sandbox’s reporting features aren’t perfect, it reports only the malicious program’s visible behavior and not how it’s programmed.

For the second viewpoint [5][6] transform follow reports into arrangements and use consecutive separations to gathering them into bunches which are accepted to compare to malware families. The principle lack was because of the unsupervised learning of bunch; it can't utilize the direction of the current infection database. [7] Presented a technique shut to their strategy, they utilized content classifications system to group obscure malware tests in view of their conduct report. Be that as it may, they overlooked the semantic relationship of the element word between variations of malwares which will likewise prompt a high measurement vector space. Highlight determination must be utilized to enhance the characterization productivity.

3. EXPERIMENT AND EVALUATION

3.1 Malware Behavior Analysis
Behavioral investigation concentrate on what projects do. The initial step of their strategy is the robotized examination of malware tests. For this reason, the authors have expanded Argus [8], their apparatus for robotized dynamic malware examination. Since malware regularly posture solid risk to the PC framework and the nearby system, so the authors partition the analyzer framework in two administrator framework. One is host and the other is guest framework. The authors use Azure cloud to comprehend this basic designing. Argus is running inside in the Azure which gives a solidly controlled course of action of advantages for activities to continue running in, for instance, framework access, virtual space on plate and memory.

The ability to inspect the host system or read from input device could be usually disallowed or heavily restricted [9]. The examination framework let malware execute in the copied The examination framework let malware execute in the imitated environment for compelled time, normally perhaps two or three minutes. Malware will summon numerous framework calls to do their vindictive activities [10]. For instance, attempting to adjust certain parts of the framework registry, or compose to pre-characterized envelopes. The activity can be blocked, or the client informed about the

endeavored activity. Consolidating API snaring and DLL infusion inside of the Azure, their analyzer instrument can follow and screen all significant framework calls amid the running of the malware.

3.2 Datasets

The authors utilized three datasets: A TrainLable, a test dataset, and a train dataset. The quantity of malware records and individually clean documents in these datasets is appeared in the initial two segments. As expressed over, their fundamental objective is to accomplish malware location with just a couple (if conceivable 0) false positives, hence the spotless documents in this dataset (furthermore in the scale-up dataset) is much bigger than the quantity of malware records. The information set comprises of malware information set, both are in the arrangement of gathering (.asm) and byte (parallel).

From the entire list of capabilities that the authors made for malware recognition, 308 double components were chosen for the investigations to be displayed in this paper. Records that produce comparative qualities for the picked list of capabilities were checked just once. Note that the quantity of clean mixes i.e. blends of highlight qualities for the spotless documents in the three datasets is much littler than the quantity of malware extraordinary mixes the spotless documents in the preparation database are mostly framework records (from distinctive forms of working frameworks) and executable and library records from diverse mainstream applications. The authors likewise utilize clean records that are stuffed or have the same structure or the same geometrical likenesses with malware documents (e.g. utilize the same packer) keeping in mind the end goal to better prepare and test the framework.

The malware documents in the preparation dataset have been taken from the Training Data Set. The test dataset contains malware documents from the TrainLable accumulation and clean records from distinctive working frameworks (different documents that the ones utilized as a part of the first database). The malware accumulation in the preparation and test datasets comprises of Ramnit, Lollipop, Kelihos_ver3, Vundo, Simda, Tracur, Kelihos_ver1, Obfuscator-ACY and Gatak sorts of malware. The primary and third sections in Table II speak to the rate of those malware sorts from the aggregate number of documents of the preparation and separately test data.

4. ALGORITHM

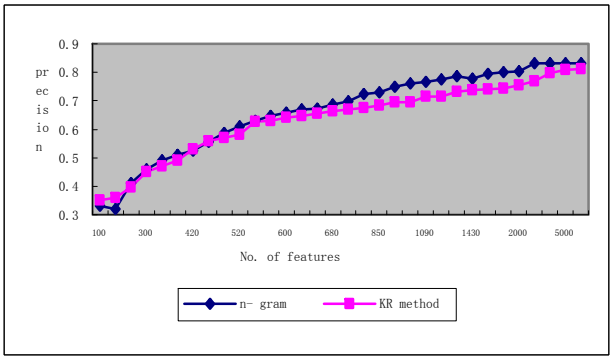
The test shows, the authors get an information set of 5231 malware tests with 9 classes from aggregate 19118.

Algorithm 1 : n-gram

```
malware<-read.csv("trainLabels.csv")
 def join_ngrams (num = 100000):
 dict_all = dict()
 for c in range(1,10):
 print "merging %i out of 9"%c
 pickle.dump(dict_all, open('ready_for_selection.pkl','wb'))
load data

#instead of binary features, do count
      grams_dict = dict()
      for gram in grams_string:
        if gram not in grams_dict:
          grams_dict[gram] = 1
        else:
          grams_dict[gram] += 1
      binary_features = []
    for feature in features_all:
      if feature in grams_dict:
        binary_features.append(grams_dict[feature])
      else:
        binary_features.append(0)
    del grams_string'''
    yield [f_id] + binary_features
  with open('train_data_750.csv','wb') as outfile:
  with open('test_data_750.csv','wb') as outfile:
    "DONE 4 gram features!"
```

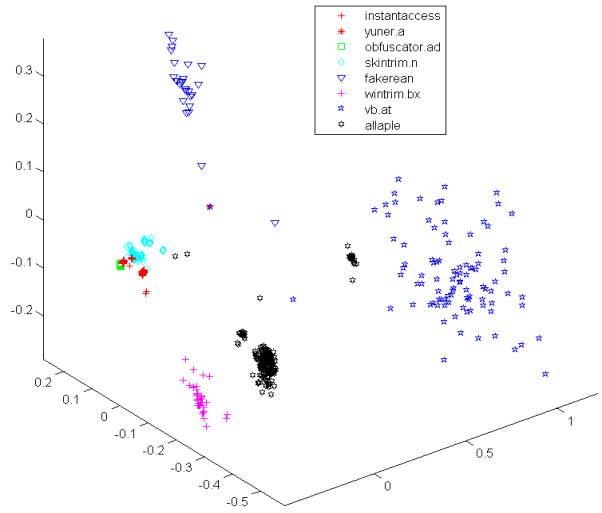
These specimens are subjectively part into two bundles are arbitrarily part into two parcels, a preparation and testing segment, preparing contains 2664 examples and test is 1332. Single-byte recurrence, byte 4-gram, direction check, capacity names and Derived Assembly Features (DAF). These components are motivated by paper and beat benchmark code in the gathering. For 4-gram bytes, the authors utilize information addition to choose the best 500 elements for every class. For direction number, regular guidelines like 'MOV' and 'JMP' are tallied. Capacity name elements should be DLL highlights. The authors supplant it by social event just the capacity names rather for effortlessness. For DAF, every element is a standard representation of asm directions, taking after the malware family: name. para1.para2, for example, or memory register. In their trial the authors pick n-gram, so the authors perform their system on their dataset. The authors can make an examination with their system. They didn't give the insights about the component vector space, so the authors receive the properties diminishment of Information-Gain [13] on the capabilities with these two technique. The authors figure the Information Gain of every component: $p(v\ Cj\ ,)$



The result of comparison

$$IG\ j(\ ) = -vj\sum \sum\in(0,1)\ c\in\{cj\ \}\ P\ v\ C(\ j\ ,)\log —p(vj\ )\ (\ )p\ C$$

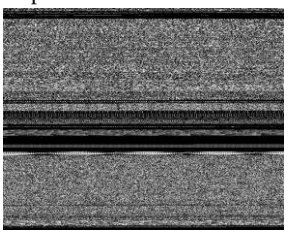
Where $IG\ j(\ )$ denote the Information Gain value of feature j , C represents one class in$\{C_i\}$, $\{C_i\}$ represent the class set of malware family. $p(v\ C_j\ ,\ )$ denotes the probability of feature j with the value of v_j in class C . $p(v_j\ )$ denotes the probability of feature j equal v_j in all training sets. $p(\ )C$ denotes the probability of class C in all training sets. At last, the authors will select features which have the lower IG value and save in the feature database [11].



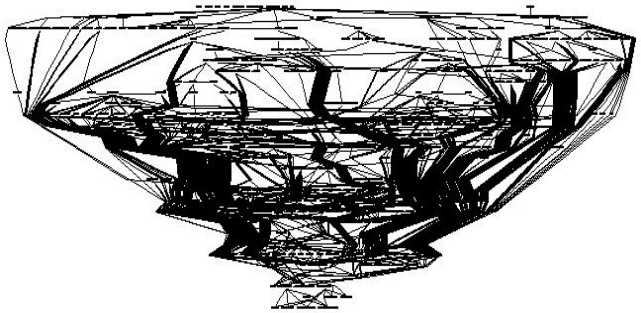
kNN - clustering of malware definitions

4.1 ASM file Pixel Intensity Feature

Malwares can be visualized as gray-scale images using the byte file [15]. Each byte is from 0 to 255 so it can be easily translated into pixel intensity. However, the authors found the image processing techniques in [15] doesn't work well with their features above. Inspired by the [14], the authors tried to extract a gray-scale image from asm file rather than the byte file. The code is shown in Figure 1. Figure 2 compares the byte image and asm image of the same malware, but this method is data driven and hence can't stop attack on same time.



Classifier	Accuracy
***k*NN (Image)**	79.4%
n-grams	85.0%
J48	96.3%



Malware graph tree obtained from IDA pro

5. CONCLUSION AND FUTURE WORK

In their paper essential target was to think about a machine learning design that insipidly perceives as much malware tests as it can. In the interim, the authors contrasted their system and other comparable methodology. Analysis demonstrate that various machine learning techniques performs well. An impediment of element malware examination taking into account API follow report is not sufficiently vigorous particularly confronted the malware which utilized numerous hostile to following innovation. As the authors would see it, malware identification by means of machine learning won't supplant the standard discovery routines utilized by hostile to infection merchants, however will come as an expansion to them.

The execution examination of 3 unique classifiers was additionally, introduced. The general best execution was accomplished by J48 utilizing the term recurrence weight without highlight choice information set, with an exactness of 96.3%. The investigation of the tests what's more, exploratory results presumed this evidence of-idea is entirely powerful and effective in classifying malware.

6. REFERENCES

[1] G. Mc Graw, G. Morrisett, "Attacking malicious code: Report to the infosec research council", IEEE Software, 17(5) , Sept 2000, pp. 3341.

[2] Wagner M. (2004). Behavior Oriented Detection of Malicious Code at Run-time. M.Sc. Thesis, Florida Institute of Technology.

[3] C.Willems, T.Holz, and F.Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. IEEE Security and Privacy, 2007, 5(2):32-39.

[4] U.Bayer, C.Kruegel, and E.Kirda. TTanalyze: A Tool for Analyzing Malware. In 15th Annual Conference of the European Institute for Computer Antivirus Research, Hamburg, Germany, 2006: 180–192.

[5] M.Bailey, J.Oberheide, J.Andersen, Z.M.Mao, F.Jahanian,and J.Nazario. Automated classification and analysis of internet malware. In Proceedings RAID07, pages 178–197,2007.

[6] Tony Lee & Jigar J. Mody Behavioral Classification. In Proceedings of EICAR2006, April 2006.

[7] T.Holz,C.Willems,K.Rieck,P.Duessel,andP.Laskov.Learning and Classification of Malware Behavior.In DIMVA08,June2008

[8] Yongtao Hu Unknown Malicious Executables Detection Based on Run-Time Behavior In Fuzzy Systems and Knowledge Discovery, 2008 pp. 391-395.

[9] Wikipedia, "Sandbox" [Online], Available: http://en.wikipedia.org/wiki/Sandbox

[10] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (1966):707–710.

[11] Hengli Zhao, Ming Xu, Ning Zheng, Jingjing Yao, Qiang Hou Malicious executables classification based on behavioral factor analysis

[12] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, August 2000.

[13] M. Chandrasekaran, V. Vidyaraman, and S. J. Upadhyaya, "Spycon: Emulating user activities to detect evasive spyware," in *IPCCC*. IEEE Computer Society, 2007, pp. 502–509.

[14] L. Natrajan, "http://sarvamblog.blogspot.com/", [ONLINE],2014

[15] L. Natrajan, S, Karthikayen, G. Jacob, and B. Manjunath, "Malware images: visualization symposium on visualization for cybersecurity. ACM, 2011, p.4

[16] Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, AIS.

[17] D. Bilar. Statistical structures: Fingerprinting malware for classification and analysis. In Blackhat, 2006

[18] L. Breiman. Bagging predictors. Mach. Learn., 24(2):123–140, Aug. 1996

[19] L. I. Kuncheva. Ensemble Methods, pages 186–229. John Wiley & Sons, Inc., 2014.