ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

Using Two Dimensional Hybrid Feature Dataset to Detect Malicious Executables

Piyush AnastaRumao

B.E Student, Department of Computer Engineering, Fr. Conceicao Rodrigues College of Engineering, Mumbai University (MU), Mumbai, Maharashtra, India

ABSTRACT: A two dimensional hybrid feature vector set is created to empower the work of detecting and deleting malicious executables. Here the two Dimensions are Hexadecimal features and DLL features of an executable. The set contains 503 n-gram hexadecimal and 28 DLL features. As attributes from both features are been combined in dataset it makes up a Hybrid dataset. This project searches to understand the relationship between this Hybrid feature set (HFS) and whether a file is malicious executable, and ultimately construct an e□cient mechanism to detect malicious executable code. Training classifier with patterns of combined vectors which occur in almost all viruses and malicious executables. Thus, making it easy to detect new viruses and malicious files unlike traditional Anti-Virus which used signature based methods. Our Model is efficient, scalable and achieves a high accuracy with low false positive rate in detecting malicious executables.

KEYWORDS: HFS, n-gram, Hexadecimal, DLL, malicious executable, dataset, classifier.

I. INTRODUCTION

Looking at current situation around us it is very obvious that the next world war is very much going to be the one fought out in cyber space. Hacking out accounts of people and misusing it or stealing out very secret information of a trading organization or tracing out military strategic posts or as simple as stealing out people's money from bank account just by few clicks via a remote device. Such attacks are no new to us now and are on an exponential growth. As current generation finds more affinity towards their smart devices such virus attacks are on rise, taking advantage of software vulnerabilities to attack host machines. Just by installing a few malicious files from remote location which looks and works quite decent on front end of the host workstation, but has deadlier backend codes which keep stealing confidential information and passing it to attackers terminal is no new miracle. Such virus files need to be effectively detected out and deleted before they harm us.

Malicious executable file (.exe) is one of the files that contain malicious code, which when executed could cause undesirable results, security breaches or damage to a system. Detection of these types of files is of great importance, and it has always been a hot research area to counter the attacks from malicious code. Masud.et.al. (2007) introduced the use of Hybrid Feature Retrieval (HFR) model, in hope to detect malicious executable e□ciently. As mentioned by Masud ,"***In the battle field of Anti-virus work, the more successful the researchers become in cracking down malicious code, the more sophisticated mal-code appear in the wild, evading all kinds of detection mechanisms.Thus, the battle between malicious code writers and researchers is virtually never-ending***". Hence, development of e□cient mechanism to detect malicious executable files is of great importance.

Using this Hybrid model as reference we have created a new more optimized model which has lowered the number of redundant data and trained classifier using malicious and non-malicious samples. Training classifier with plenty patterns of combined vectors comprising of frequently occurring Binary n-gram selected from hexadecimal extraction and important DLL call referenced extracted from executable file which occur in almost all viruses and malicious files. Any new virus sample can be easily detected using this training unlike traditional Anti-Virus which used signature based methods which needs regular updates about new Virus definition (Check Figure 5 and Figure 6 for more detailed study of our model).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

II. RELATED WORK

In [1] authors have introduced new data mining approach to detect malicious executable rather than traditional heuristic approach which was more costly and ineffective. Using data mining framework they found out patterns which detected malicious binaries. Thus doubling the detection rate compared to traditional signature based approach. [2] We are introduced with static techniques to detect malware using techniques of information gain (IG) and principle component analysis (PCA). In [3] authors have described a hybrid model to detect malicious executable, where they have used three feature set to create a dataset. They are as follows:-i] Binary n-gram, ii] Derived Assembly Features (DAF), iii] Dynamic Link Library (DLL) calls. They use this features to train classifier and then trained classifier is used to test on unseen malicious executables. Here in [4] authors have combined machine learning techniques with data mining techniques. They have gathered numerous training samples and processed it. After this is done they have selected only relevant features for prediction and evaluated variety of methods like Naïve Bayes, Boosted decision tree. In [5] authors use techniques of sliding window to select n-grams as per the value of n taken. We select n-gram of malware instances to classify it correctly. They have mentioned how to reduce large data size and dimensions.

III. PROBLEM STATEMENT

A. *Drawbacks of Current Approaches:*

"Traditional approach" is Signature based which requires signatures to be generated by human experts. So, it is not effective against "zero day" attacks. Signature engines report many false positives. Since the appearance of the first computer virus in 1986, a significant number of new viruses have appeared every year. This number is growing and it threatens to outpace the manual effort by anti-virus experts in designing solutions for detecting them and removing them from the system. Even without this threat, the traditional approach consists of waiting for a number of computers to be infected, detecting the virus, designing a solution, and delivering and deploying the solution. A significant damage is done during this process.

B. *Our Solution To Above Problem:*

We propose our novel two dimensional model that can detect malicious executables efficiently. This is an extension to the previous work (Masud et al. 2007) [3]. As per their work they extract three different kinds of features from the executables at different levels of abstraction and combines them into one feature set, called the hybrid feature set (HFS). These features are: (a) binary n-gram features, (b) derived assembly features (DAFs), and (c) dynamic link library (DLL) call features. Here each Derived Assembly feature (DAF) is a sequence of assembly instructions in an executable, and corresponds to one binary n-gram feature. So we introduce our technique to reduce the trade-off between complexity, computing time and accuracy. We propose an efficient technique which uses two features instead of three.

We extracts two different kinds of features from the executables and combines them into one feature set. These features are used to train a classifier (e.g. support vector machine (SVM), Logistic regression, Random Forest or Boosted decision tree) [4], which is applied to detect malicious executables. These features are: (a) binary n-gram features, (b) dynamic link library (DLL) call features. Each binary n-gram feature is actually a sequence of n consecutive bytes in a binary executable. Each DLL call feature actually corresponds to a DLL function call in an executable. We show that the combination of these two features is always better than any single feature in terms of classification accuracy and speed for detection of nature of file under scan. Our work aims on increasing features at different dimensions, rather than using more features in a single dimension. There are three main reasons behind this. First, the number of features at a given level (e.g. binary) is overwhelmingly large. [5] For example, in our larger dataset, we obtain millions of binary n-gram features. Training with this large number of features is way beyond the capabilities of any practical classifier. That is why we limit the number of features at a given level of abstraction to an applicable range. Second, we empirically observe the benefit of adding more levels of abstraction to the combined feature set. Third, we optimize abstraction level from three to two thus increasing execution speed yet maintaining accuracy level [3]. It combines features at two level of abstraction, namely, binary executables and system API calls.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

We show that this combination has higher detection accuracy and lower false alarm rate than the features at any single level of abstraction. First, we use hex-dump to extract the hexadecimal features from the malicious as well as non-malicious samples and then we extract the DLL from each of the files. Second, we apply the statistical knowledge obtained during feature extraction to select the best features, and to build a classification model. This is accomplished by extracting all possible binary n-grams from the training data, applying the statistical knowledge corresponding to each n-gram (i.e., its frequency in malicious and non-malicious executables) to compute its information gain and selecting the best of them. Finally, we apply another statistical knowledge (presence/absence of a feature in an executable) obtained from the feature extraction process to train classifiers (Check Figure 5).

Going back to Research by Masud [3], two important aspects recognized as e□ciency are 1) e□ciency in extracting hybrid features, and 2) classification model. The process of extracting these hybrid features is not trivial, but involves a lot of technical challenges. To illustrate one of the di□culties, among the three categories of hybrid features, [5] the total time for collecting n-grams would be O(N^2), where N is the total number of n-grams. At the meantime, pointed out by Masudet. al., most of the features would be noisy, redundant or irrelevant. Thus, along with the desire of building a working classification rule, understanding the selection of a small, relevant and useful feature set will be the most desirable.

In our model , important features are already brought down to a hundred's level, at the time of building desirable classification model, it is also of this project's interest to understand if the number of necessary features could be further lowered.Our research contributions are as follows. Firstly, we propose and implement our two-dimensional feature set model, which combines two kinds of features as mentioned above. Secondly, we increase computational speed by reducing Dimensions of feature set as proposed by Masud.et.al. 2007 from three to two without making any trade-off in accuracy of model.

IV. MODULE DESCRIPTION

A. *Feature Extraction:-*

1. Hex Dump:-

In computing, Hex Dump is a hexadecimal view of computer data. It is commonly done as a part of Reverse Engineering. In hex dump, each byte (8 bits) is represented as 2 digit hexadecimal number. Hex dumps are organized into rows of 8 or 16 bytes, sometimes separated by whitespaces. Some hex dumps have the hexadecimal memory address at the beginning or a checksum byte at the end of each line. First step in our module is extracting the hex dump data from the training or testing executables. Hex dump data is as follows in Figure 1.

```
run:
getCurrentDirectory(): E:\try\Try2
getSelectedFile() : E:\try\Try2\Sample4509.exe


[OFFSET] [..............HexaDecimal Values.............]  [ASCII Representation]

 0000    00 6D 00 75 00 73 00 74 00 20 00 66 00 6F 00 72    .m.u.s.t. .f.o.r
 0010    00 20 00 6C 00 6F 00 63 00 61 00 74 00 69 00 6F    . .l.o.c.a.t.i.o
 0020    00 6E 00 20 00 61 00 20 00 74 00 68 00 69 00 73    .n. .a. .t.h.i.s
 0030    00 20 00 70 00 61 00 63 00 6B 00 61 00 67 00 65    . .p.a.c.k.a.g.e
```

Fig.1. Hexadecimal view of data extracted from executable file Sample4509.exe.

2. Binary N-gram:-

An n-gram is contiguous sequence of n items from a given sequence of text or speech. An n-gram of size 1 is referred to as "unigram", size 2 as a "bigram" or "diagram", size 3 as "trigram" while larger size are referred to by value of n as "four gram", "five gram" and so on. An n-Gram is a type of probabilistic language model used in probability, communication theory, computational linguistics, computational biology and data compression. Two benefits of n-

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

gram are [A] Simplicity [B] Scalability. We convert a sequence of items to set of n-grams and embed it into vector space. Once we are done with Hexdumpdata extraction from an executable file we are left with amount of Hex data which is computationally infeasible [5]. So we have deployed n-Gram selection Algorithm. Here we specify n-Gram count starting from $1 < n < 10$. Based on this we use information gain to select best most frequently occurring features from extracted hex dump data. This reduces computational cost on large scale and saves memory.Here Figure 2 is a snapshot of n-gram data from hex dump data.

```
Sample4509.exe
getCurrentDirectory(): E:\try\Try2
getSelectedFile() : E:\try\Try2\Sample4509-HexDump.hex

5F 00 70 00
74 00 20 4A
74 00 20 4C
74 00 20 4D
FF FF 20 00
CC CC CC 3F
7F 74 A3 33
00 54 4C 4F
3A 30 3A 34
74 00 20 43
CC CC CC 3B
```

Fig.2. Binary n-gram extraction from large Hex dump data File of executable Sample4509.exe.

3. Dynamic Link Library (DLL):-

A DLL is a library that contains code and data that can be used by more than one program at the same time. For example, in Windows operating systems, the Comdlg32 DLL performs common dialog box related functions. Therefore, each program can use the functionality that is contained in this DLL to implement an Open dialog box. This helps in code reuse and efficient memory usage. DLL reduces memory space by optimizing RAM however there always exists a threat where original DLL is replaced with a fake DLL thus leading to DLL hijacking.We use here DLL function call at granular level. We extract the information about DLL function calls made by a program from the header of the file. We collect all the DLL function names that have been used by each of the benign and malicious executables, and select the best most frequent and commonly occurring S of them using information gain. Here Figure 3 is a screenshot of the sample DLL function calls that we extract from an executable header.

```
advapi32
user32:EndDialog()
WINNM.DLL
CntrtextInstaller.DLL
ADVAPI32.DLL:AdjustTokenPrivileges()
USER32.DLL
AVICAP32..DLL
ADVAPI32.DLL
```

Fig.3. DLL function calls extracted from an executable file.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

B. ***Feature Selection:-***

The total number of extracted features is very large, it is not possible to use all of them for training because of several reasons. First, the memory requirement may be impractical. Second, training may be too slow. Third, a classifier may become confused with a large number of features, because most of them would be noisy, redundant or irrelevant. So, we are to choose a small, relevant and useful subset of features. [2] We choose information gain (IG) as the selection criterion, because it is one of the best criteria used in literature for selecting the best features. IG can be defined as a measure of effectiveness of an attribute (i.e., feature) in classifying a training data (Mitchell 1997). If we split the training data based on the values of this attribute, then IG gives the measurement of the expected reduction in entropy after the split[6]. The more an attribute can reduce entropy in the training data, the better the attribute is in classifying the data. So we have to select only the best and most frequently occurring features.

V. CREATING A HYBRID 2-DIMENSION VECTOR FEATURE SET

Once we are done with extraction and selection step, where we select hexadecimal binary n-gram and DLL features of a single .exe. We have two independent set of features available. This features won't be useful for best results if used independently as it won't give a consistent pattern. So we have to combine this two vectors of binary n-gram and DLL feature set into a higher dimensional feature set. Hybrid vector space with higher and combined attributes can then be used for further calculations for getting a better result. Figure.4. clearly depicts how hybridization is been done by combining two independent features- a] N-Gram features extracted from Hex dump and b] DLL call features from the header of executables. Though we apply a filter before this hybridization phase selecting only most relevant and frequently occurring features from either side. Thus lowering the memory requirement and scanning out features which won't have considerable impact on detecting the malicious sample. Our Hybrid file created as per Figure.4. consist of data in following sequence 1] N-Gram followed by 2] DLL Calls. Even though the sequencing of hybrid file data has no such impact on detection accuracy of the module, yet we maintain a standard sequence throughout the module.

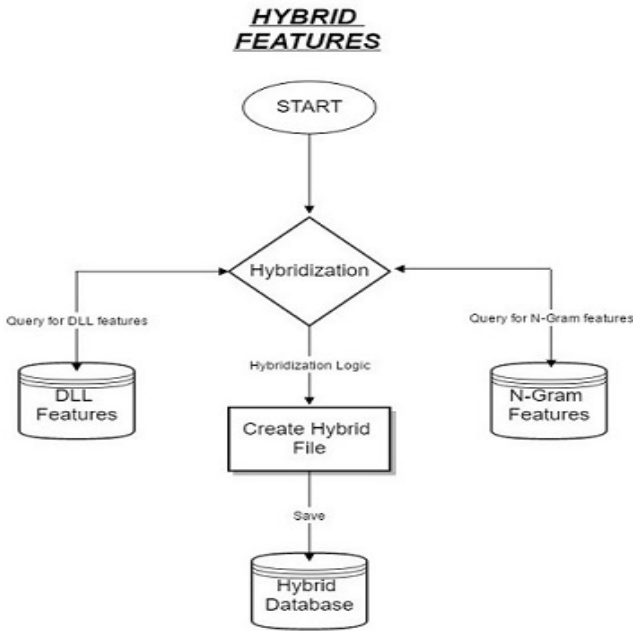

Fig.4. Hybrid Feature Dataset Generation.

Steps involved:-Figure.5 and Figure.6 will give us a clear picture of working of our module which is broadly classified into two phases- A] Training phase and B] Testing phase. The graphical flowchart clearly shows how each step is been executed in a sequence one after one other. Finally we have Testing phase in Figure.6 which follows same steps like

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

training with additional steps which detects the nature of test sample based on past training file and deletes the sample if tested positive for malware.

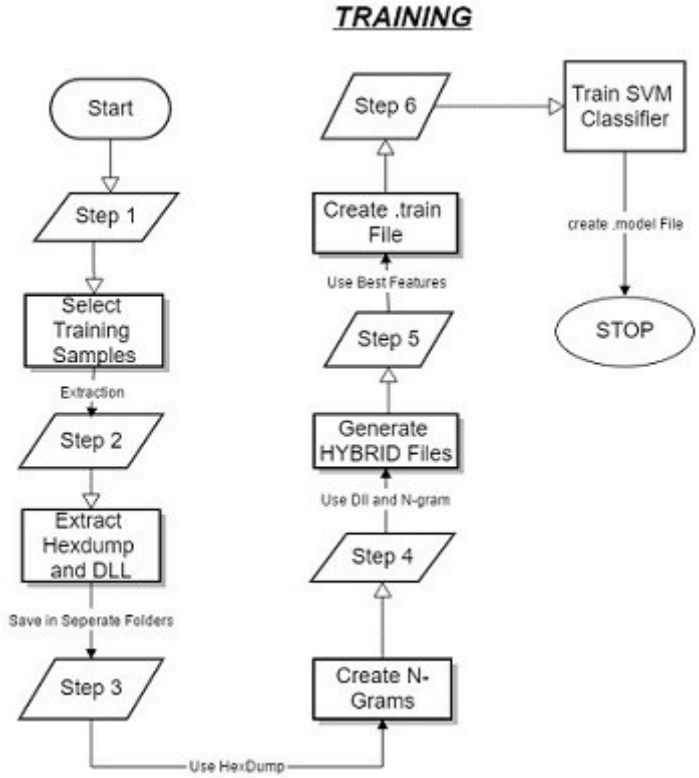

Fig.5. Sub steps involved in training phase.

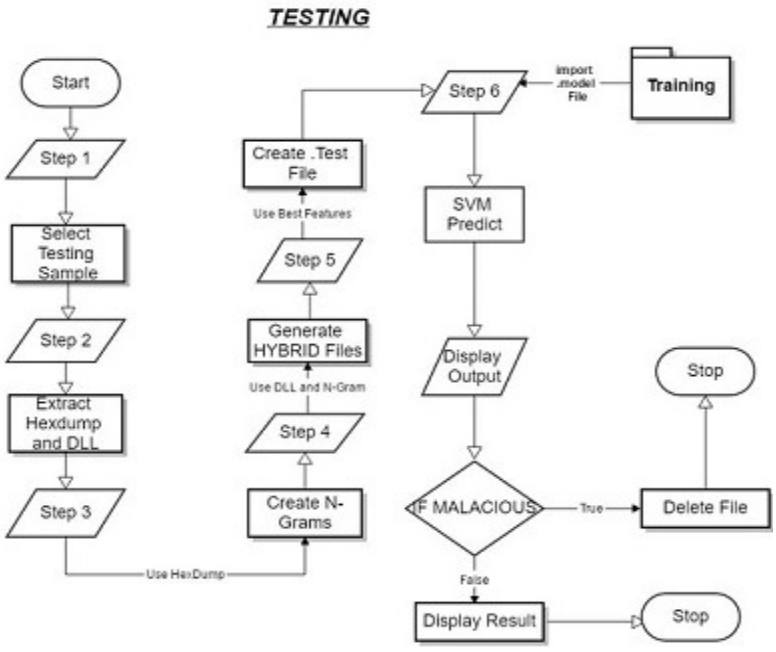

Fig.6. Sub steps involved in testing phase.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

VI. DATASET DESCRIPTION

We have two independent datasetsone for training our classifier while other for testing our trained classifier.Training dataset is saved with a .train extension while one for testing unknown data is saved with .test extension. Our First dataset contains 2400 executables, of which 800 are non-malicious executables while 1600 malicious executables.While dataset 2 has 373 executables having 100 non-malicious while 273 malicious executables. So dataset-1 has 33% non-malicious executables and 67% malicious executables and that of dataset-2 is 26% non-malicious executables while 74 % malicious executables.

2.1 Raw Data Cleaning

Raw data came in the form that for every .exe file, the record starts with label with level 1/-1, with 1=Non-Malicious executable and -1=Malicious executable. A record will end with closing mark, which is a number -1. Between the record label, and closing mark, there are feature existence identifiers. For example, 53:1 represents that the 53th feature exists in this .exe file.[10]Clean data contains 532 columns, with the 1st column recording label information, 1/-1. And the rest 531 columns record 531 binary variables, with 1 denoting the existence of that feature, and 0 otherwise.However as we are using sparse matrix so the feature which is absent for a particular sample instead of showing a 0 in front of it we simply ignore it and print only 1 for those features which are present.The clean data set contains 373 rows, each stores information for an executable file. Figure 7 is a snapshot of one of our dataset.

```
+1 1:1 3:1 5:1 7:1 9:1 11:1 15:1 19:1 26:1 30:1 33:1 37:1 39:1 42:1 44:1
+1 1:1 3:1 5:1 7:1 9:1 11:1 15:1 19:1 26:1 29:1 33:1 37:1 39:1 42:1 44:1
+1 1:1 3:1 5:1 7:1 9:1 11:1 15:1 19:1 21:1 24:1 26:1 29:1 31:1 33:1 36:1
-1 1:1 3:1 5:1 7:1 9:1 11:1 20:1 26:1 30:1 33:1 37:1 39:1 42:1 44:1 49:1
-1 1:1 3:1 5:1 7:1 9:1 11:1 15:1 20:1 26:1 30:1 37:1 39:1 42:1 44:1 49:1
```

Fig.7. Snapshot of Hybrid Feature Dataset Generated.

Where starting column with values of +1 and -1 are labels indicating non-malicious or malicious executables whereas rest are features which exists in that sample indicated with a feature number and :1 indicating that feature exists.We can apply SVM, Random forest, logistic Regression, Boosted decision tree, and other classifiers for the classification task on this dataset.

2.2 Data Distribution and Dependence Structure

Let us consider 373 .exe files, and among them 72 are non-malicious executable and 301 are malicious executable. In total 531 hybrid features are extracted:

- 503 hex decimal features: feature index from 1 to 503;
- 28 DLL features: feature index from 504 to 531.

Note:- Our sample dataset is been published on **The University of California, Irvine (**UCI Machine Learning Repository: Data Sets) and can be found directly on following link https://archive.ics.uci.edu/ml/datasets/Detect+Malacious+Executable(AntiVirus)

VII. PERFORMANCE EVALUATION

With the importance to develop an e□cient malicious detection mechanism in mind, this paper considers following items as important performance evaluation criteria for constructing classification models.

1. **Accuracy Rate from Cross-validation**
   Running penalized logistic regression, random forest and SVM models for 100 times, each time taking 20% of data as test, and the rest 80% as training set. Comparing the predicted labels with the truth, the accuracy rate will be calculated. Figure 8 and Figure 9 presents the results. In Figure.8 we see that cross validation results for Random forest with two sets- A] All features used and B] top 20 features used results in a highly optimized

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

model due to optimized hybrid file created during training.Surprisingly, we find that penalized logistic Regression too offer us with similar output as Random forest.

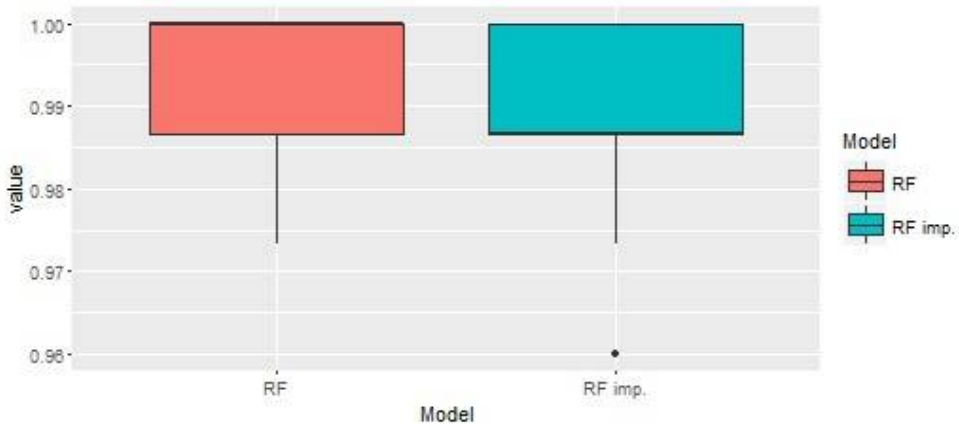

Fig.8. Cross-validation results for Random Forests: 1)RF: all features used, 2) RF. Imp: Top 20 Features.

In Figure.9 we use LIBSVM classifier to check the accuracy of our module using SVM classifier. After running cross-validation phase on sample data we calculate the accuracy of our module each time offering us an accuracy in the range of 90-96%. Figure.9 shows a snapshot of one such random run using LIBSVM java code.

```
* Libsvm classifier
optimization finished, #iter = 74
nu = 0.119053929248003
obj = -25.79939960623191, rho = -1.00860458123023
nSV = 60, nBSV = 31
Total nSV = 60

...Testing...
Accuracy = 93.0%  (classification)
E:\try
Firstline is : -1.0
malacious file
BUILD SUCCESSFUL (total time: 3 minutes 27 seconds)
```

Fig.9. Accuracy Result for SVM using LIBSVM.

2. Computing Time

The computing time is obtained from System.time() function in R and Netbeans System Runtime for LIBSVM that calculates the real time user would encounter. Logistic regression is the winner in computing time to construct classification model. Random forest as mentioned earlier involves twice the computing time in terms of building a forest, since it involves two steps, 1) selecting important features from a forest, and 2) build the forest using selected features.While SVM too consumes time between the two but giving a higher accuracy.

LogitR→65.55 sec RF→ 209.4 sec SVM→200 sec

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

3. Stability in feature selection

Running both penalized logistic regression and random forest for 100 times, a list of features that are selected in each time is recorded. This features are among the best features selected in our hybrid file. To be specific 531 best features. Lasso is used as the regularization tool for logistic regression, for the fact that this data set all features are highly relevant in terms of predicting malicious executable, it is very likely a lot of the features are correlated. And lasso is not good in handling correlated predictors, and will be unstable when selecting predictors. Random forest is relatively more stable compared to lasso. In the 100 iterations of model building, random forests in total selected 41 unique important features (20 every time), 50% of them have high frequency of appearing in most of the iterations, while penalized logistic regression has only 25% of these features (see Figure 10 for further details).

Logistic		RandF	
Features	**Count**	**Features**	**Count**
v19	100	v19	100
v20	100	v66	100
v139	88	v67	100
v447	87	v139	100
v140	83	v140	100
v403	74	v20	99
v266	70	v51	98
v291	70	v162	97
v179	67	v111	96
v69	56	v179	96
v435	31	v385	91
v182	27	v113	89
v47	23	v266	86
v162	23	v69	81
v53	19	v334	80
v26	16	v278	76
v385	16	v50	64
v66	11	v49	59
v111	10	v182	54

Fig.10. Most Commonly Appeared Features during 100 times of Model Iterations.

4. Complexity to Interpret

Random forest is constructed by randomly selecting m features among available ones, and construct trees in every selection. The randomness of such method results in the fact that it will be hard to have a written-down model that one can refer to for future work. Being a black box is a draw-back of random forest. SVM on other hand stands much ahead of this both with its linear, non-linear kernels as well as hard margin and soft margin features. However it requires full labelling of input dataset.

VIII. CONCLUSION

We combined n-gram Hexadecimal features with DLL function call to create a two dimensional scalable Hybrid feature dataset and used it effectively to detect malicious executables. We have addressed numerous issues here like efficiency, speed, memory size and accuracy. Though still a detailed analysis and performance test can be conducted for better results. More classifier algorithms with range of parameters need to be tested out. We still face problems in our training phase execution time. The Hexadecimal feature extraction time is exponentially high which needs to be yet

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

brought down for quicker training as well as testing. Our model is for now restricted to Desktop applications and cannot be implemented online or to detect potentially harmful websites that our users visit, thoughwe will address this issues in our future work.

REFERENCES

1. Matthew G. Schultz, ErezZadok, SalvotoreJ.Stolfo and EleazarEskin, "Data Mining Methods for Detection of New Malicious Executables", Security and Privacy, S&P Proceedings. 2001 IEEE Symposium, SP'01, 38, May 2001.
2. UsukhbayarBaldangombo, NyamjavJambaljav, and Shi-Jinn Horng, "A STATIC MALWARE DETECTION SYSTEM USING DATA MINING METHODS", International Journal of Artificial Intelligence & Applications (IJAIA), Vol. 4, No. 4, July 2013.
3. MM Masud, Latifur Khan and BhavaniThuraisingham. "A Hybrid Model to Detect Malicious Executable (Using Data Mining and Machine Learning Concept)".IEEE Communications Society, June 2007.
4. J. Zico Kolter, Marcus A. Maloof, "Learning to Detect and Classify Malicious Executables in the Wild", Journal of Machine Learning Research, Vol. 7, pp.2721-2744, January 2006.
5. Abdurrahman Pektaş, Mehmet Eriş and TankutAcarman,"Proposal of n-gram Based Algorithm for Malware Classification", International Academy,Research and Industry Association(IARIA), 2011.
6. Mohammad M. Masud, Latifur Khan and BhavaniThuraisingham, "A hybrid Model to Detect Malicious Executable", IEEE International Conference on Communications, June 2007.
7. Tony Abou-Assaleh, Nick Cercone, VladoKeˇselj and Ray Sweidan. "N-gram-based Detection of New Malicious Code", Computer Software and Applications Conference, Vol. 2, pp 41-42, September 2004.
8. LIBSVM. (2006). A library for support vector machine. Retrieved June 1, 2006 from http://www.csie.ntu.edu.tw/~cjlin/libsvm/.
9. VX-Heavens. (2006). Retrieved May 6, 2006 from http://vx.netlux. org/.
10. Piyush Rumao, "Anti-Virus Dataset", March 2016, published on the website of University of California, Irvine (UCIMachineLearning Repository:DataSets)and can be found by clicking the following link https://archive.ics.uci.edu/ml/datasets/Detect+Malacious+Executable(AntiVirus).

BIOGRAPHY

Piyush Anasta Rumao has done his **Bachelor of Engineering** (B.E) in Computer Science from Fr. Conceicao Rodrigues College of Engineering, University of Mumbai, Mumbai, India. His research interests include **Machine Learning, Big Data and Security**. He has published an Anti-Virus **dataset** on Machine Learning Repository website of **University of California, Irvine** and is currently working on numerous projects including **Security and Privacy in Internet of things (IOT)** and creating **an e-commerce website for ordering Study materials and Books.**