Implementing Machine Learning Techniques in Malware Detection

Adnaan Arbaaz Ahmed[1,*], D.Rajesh[2], Dr.M.I.Thariq Hussan[3]

[1,2]B.Tech Scholar, [3]Professor & Head, Department of Information Technology, Guru Nanak Institutions Technical Campus, Hyderabad, India

***Corresponding Author: adnaanarbaazahmed14141114@gmail.com**

Abstract - Malware is compatible in a world of innumerable computer users, who are constantly faced with these threats from various sources like the internet, local networks and portable drives. Malware is more probable low to high risk and can cause systems to function incorrectly, retrieve data and even crash. Malware may be executable or system library files in the form of viruses, worms, Trojans, all targeted at exploiting the security and performance of the system and compromising user's data privacy. Typically, anti-virus software is based on a signature definition system which keeps updating from the internet and thus keeping track of known viruses and defends them. In this paper, we present a new, improved and more sophisticated antivirus engine that can scan files, build knowledge and detect files as potential viruses. Here we introduce a versatile framework in which we can employ various machine learning algorithms to transparently distinguish between malware files and clean files by targeting to minimize the number of myths. We also present the ideas behind our framework by working primarily with cascade one-sided perceptron's and secondarily with kernelized one-sided perceptron's.

Keywords - Data Mining, Datasets, Firewall, Machine Learning, Malware Detection, Network Security, One-sided perceptron's, Random Forest Algorithm, Virus

I. INTRODUCTION

Malware refers to malicious software designed to infiltrate or damage a computer system without the owner's consent. It is a generic definition for all computer threats. It usually consists of programming code, scripts or other content designed to disrupt operation or gather information that leads to loss of privacy, gain unauthorized access to system resources and other abusive behaviour [8]. There are two different classifications of malware. A simple classification consists of file infectors and stand alone malware. Other classification is based on their particular action such as worms, backdoors, Trojans, viruses, adware, spyware, ransomware etc.,[6]. Any software is classified as malware based on the intent of the maker instead of any particular feature. According to F-Secure, much malware produced in 2016 as in the previous 20 years altogether. With the rise in widespread internet access, malicious software has been designed for a profit. The majority of these viruses are meant to take complete authorities and command over user's computers and data. Spyware are programmed to monitor users web browsing and extract personal information. Unlike virus spyware are programmed not to spread, but are installed to exploit security packages or packaged with user software. transparently, there is a rampant need to find, else not just a required method to detect virus probed files, but to build a effective mechanism that can detect virus by studying the structure of system responses made by malware.

Machine learning is a branch of Artificial Intelligence knowledge concerned with the design and development of algorithms that allow computers to imbibe behaviors based on structured data, such as from sensor or through database action can take advantage of data to know characteristics of interest of their unknown underlying probability distribution [4]. A major focus of the investigation in machine learning is to learn to identify difficult or complex patterns and to automatically make intelligent decisions based on the database [11].

II. CURRENT ANTI-VIRUS ENGINES

Antivirus is used to prevent, detect, and remove malicious malware, virus, computer worm, Trojan horses, spyware and adware. A number of strategies are strategically employed by the anti-virus engines. Signature based investigation involves searching for known patterns of data within executable code [14]. Today, malware is no longer an executable file. Powerful macros in Microsoft Word could also be at security risk. A variety of strategies are systematically employed by the anti-virus engines. Signature based detection includes searching for known patterns of data within executable code. However, it is possible for a computer to identify the infected with new virus for which no signatures exist [5][15]. To counter such "zero-day" threats a heuristics can be used, that detects new viruses or family of existing viruses by looking for known malicious code. Some antivirus makes predictions by executing the code in the sand box. Often, antivirus software can affect computer's performance. Any false decision may lead to a security leak and data breach, since it maintains at the highly trusted kernel level of the operating system. If the antivirus software employs extreme detection, success depends on the correct middle point between false

 DOI : 10.18231/2454-9150.2018.1173 © 2018, IJREAM All Rights Reserved.

positives and negatives. Traditionally, antivirus is a signature to identify malware [2][18]. However, because of newer methods of creation of malware, signature-based approaches are no longer effective.

III. OUR APPROACH

The current antivirus engine performances are not optimum in detecting viruses to the maximum. They may be useful in virus control once they affect systems [6], which fateful for enterprises. This research is a firewall level of enterprise network.

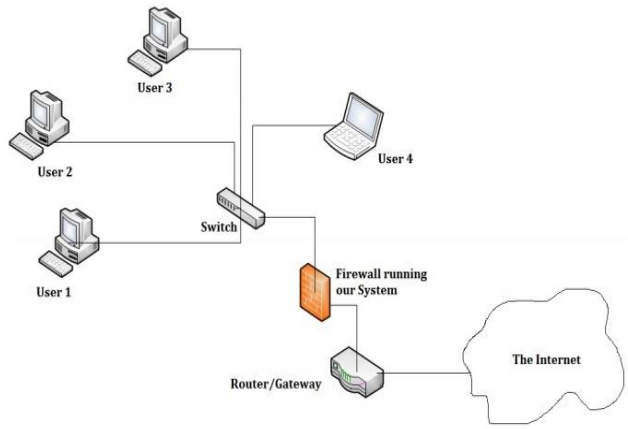


Figure 1: Network diagram of entire system

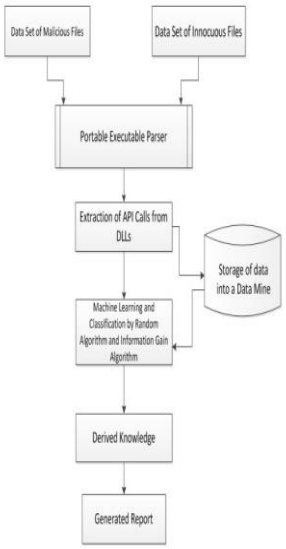


Figure 2: Process diagram of our system

A. Information Gain

It is important to calculate the information which is generally expressed by the average number of bits needed to store or communicate single symbol in a message. It has quantified in quantity [1].

Information theory measure information in bits

Entropy(p1,p2,….pn)=-p1log(p1)-p2log(p2)-……..-pnlog(pn)

Information gain is the amount of information gained by expressing the value of the attribute

Information gain=(Entropy of distribution before the split)-(entropy of distribution after split) [10].

Information weighs the amount of information that is acquired by knowing the value of the attribute, which is the entropy of the distribution before it splits minus the entropy of the distribution after it. Higher the information gain smaller the entropy.

Further, we illustrate the Random Forest Algorithm (RFA). This is a machine learning classification algorithm which constructs the classifier to detect malware. Random Forest is one of the most effective machine learning algorithms. It requires almost no data entity and modelling but generally ends in accurate results. More speculatively, Random Forests are the collections of decision trees, producing a better prediction rate [9]. That's the reason why it is called a 'forest' which is basically a set of decision trees. The basic model is to grow multiple decision tree based on the independent subsets of the dataset. At each node, n variables from the feature set are selected randomly, and the best split on these variables is found. Generally, the algorithm can be described as follows

Step 1. Many trees are built roughly on the quarter of the training data (62.3%). Data is chosen randomly.

Step 2. Several predictor attributes are randomly picked out of all the predictor variables. Then, the optimal split on these selected variables is used to split the node. By default, the amount of the selected attributes is the square root of the total number of all predictors for classification, and it is the same for all trees.

Step 3. Using the remains of the data, the misclassification rate is calculated. The aggregate error rate is calculated as the overall out-of-bag error rate.

Step 4. Each trained tree gives its own classified result by specifying its own "vote". The class which receives the more number of votes is chosen as the result.

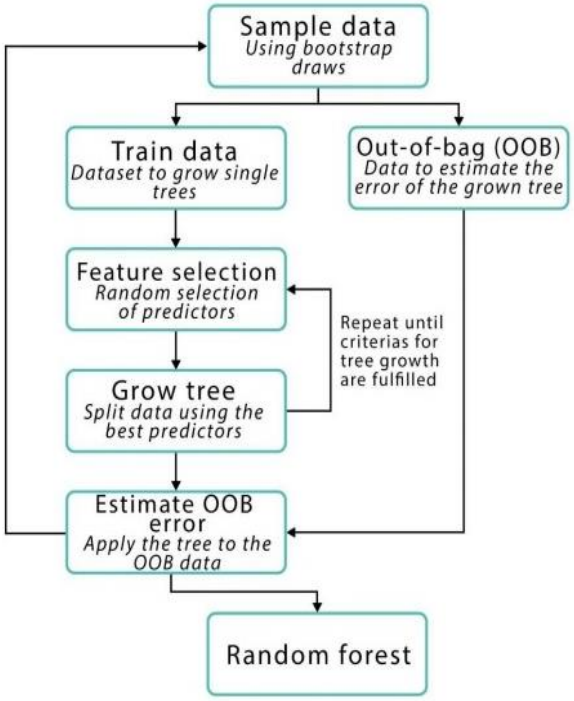


Figure 3: Random Forest Scheme

As in the decision trees, this algorithm eliminates the need for feature selection for removing irrelevant features – they

 DOI : 10.18231/2454-9150.2018.1173 © 2018, IJREAM All Rights Reserved.

will not be taken into consideration in any case. The main need for any feature selection with the random forest algorithms arises when there is a need for measuring reduction. Moreover, the explicit error rate, which was structured earlier, can be considered the algorithm's own verification cross-validation method. This eliminates the requirement for unwanted or tedious cross-validation measures, that would have to be taken or else, Random forests will inherit many of the advantages of the decision trees algorithms. They are applicable to regression and classification problems; they are easy to compute and ease to fit. They also generally result in the better prediction and accuracy. In decision trees, by evaluating the resulting tree, we can gain generic information about which variables are important and how they affect the result. It is impossible with random forests. It can also be narrated as a more stable algorithm than the decision trees – if we modify the data a little, decision trees will change, most likely reducing the prediction and accuracy. This will not happen in the random forest algorithms – since it is the blend of many decision trees, the random forest will remain constant.

B. Datasets

Here we have used three datasets: a training dataset, a test dataset, and a scale-up dataset. The number of malware files and clean files respectively in these datasets are shown in the first two columns of Table I. As stated above, our main goal is to achieve malware detection with only a few (possibly 0), a false positive, therefore the clean files in this dataset (and also in the scale-up dataset) is larger than the number of malware files.

TABLE I
NUMBER OF FILES AND UNIQUE COMBINATIONS OF FEATURE VALUES IN THE TRAINING, TEST, AND SCALE-UP DATASETS.

	Files		Unique combinations	
Database	malware	clean	malware	clean
Training	27475	273133	7822	415
Test	11605	6522	506	130
Scale-up	approx. 3M	approx. 180M	12817	16437

The malware files in the training dataset are taken from the Virus Heaven collection. The test dataset contains malware files from the WildList collection and clean files from various operating systems (other files that the ones used in the first database). [16]The malware collection in the training and test datasets consists of trojans, backdoors, hacktools, rootkits, worms and other types of malware. The first and third columns in Table II represent the percentage of those malware files from the total number of files of the training and test datasets respectively. The second column in Table II represents the proportionate percentage of malware unique combinations from the total number of unique combinations of feature values for the training dataset. As shown in the first and lasts column of Table II, the distribution of malware files in this test dataset is significantly different from the malware distribution in the training dataset. The third, large dataset was used for testing the scaling up capabilities of our learning algorithms. This dataset was divided into 10 parts denoted as S10, S20, .., S100, where Si represents i% of the total dataset, and Si ⊂ Si+10. We have used these parts in order to evaluate the training speed and the malware detection rate for larger datasets.

IV. ALGORITHMS

The main goal of this section is to modify the perception, so as to correctly detect malware files, while forcing (if possible).

Algorithm 1 The Perceptron Training Subroutine

```
Sub Train (R, LR_Malware, LR_Clean) :
γ_i = 0
i = 1, . . . , n
for all record in R do
  if Fitness(record) ≠ record.label then
    for all γ_i do
      if record.F_i ≠ 0 then
        if record.label = 1 then
          γ_i = γ_i + LR_Malware
        else
          γ_i = γ_i + LR_Clean
        end if
      end if
    end for
  end if
end for
for all w_i do
  w_i = w_i + γ_i
end for
End sub
```

Algorithm 1 is the standard perceptron algorithm. Instead of working with floats, it uses a large integer delineation for the weights w_i, i = 1 . . . n, where n is the total number of attributes/features. This adaptation is non-restrictive, guessing multiplication with a certain factor related to the representation.

Other notations used by this algorithm are:

− γ_i, i = 1 . . . n, are the additive values that will modify the weights wi after one repetation.

− LR Malware and LR_Clean are the learning rate constants for the malware and the clean sets of files respectively.

The perceptron's fitness function is defined as: Fitness (R_i) = Sign(Pn_j=0 $w_j R_i.F_j$ − Threshold),

where $R_i.F_j$ denotes the value of the feature fa_j in the file record R_i .

We define Sign(x) = 1 if x ≥ 0, and -1 otherwise.

We used the same one-sided perceptron, but in the dual form and with the training entry mapped into a larger feature space via a kernel function [18]. The resulting kernelized one-sided perceptron.

TABLE II
MALWARE DISTRIBUTION IN THE TRAINING AND TEST DATASETS.

	Training Dataset		Test Dataset
Malware type	Files	Unique combinations of feature values	Files
Backdoor	35.52%	40.19%	9.16%
Hacktool	1.53%	1.73%	0.00%
Rootkit	0.09%	0.15%	0.04%
Trojan	48.06%	43.15%	37.17%
Worm	12.61%	12.11%	33.36%
Other malware	2.19%	2.66%	20.26%

 DOI : 10.18231/2454-9150.2018.1173 © 2018, IJREAM All Rights Reserved.

Algorithm 2 Kernelized One-Sided Perceptron

```
for i = 1 to n do
    Δ_i ← 0
    α_i ← 0
end for
for i = 1 to n do
    if (label_i × Σ_{j=1}^{n} (α_j × K(i,j))) ≤ 0 then
        Δ_i ← Δ_i + label_i
    end if
end for
for i = 1 to n do
    α_i ← α_i + Δ_i
    Δ_i ← 0
end for
```

The algorithms 1 and 2 presented above will be used in the sequel as bricks in cascade classification algorithm. Given a set of binary classification algorithms {A1,A2, . . . ,Ak}, a cascade over them is an aggregated classification algorithm that classifies a given test instance is as follows.

Algorithm 3 Cascade Classification

```
if A_{i1}(x) or A_{i2}(x) or ... A_{ik}(x) then
    return 1
else
    return -1
end if
```

V. CONCLUSION

In this research, we have proposed a malware detection module based on modern data mining and machine learning. While such a method may not be apt for home users, being very processor heavy, this can be accomplished at enterprise gateway level to act as a central antivirus engine to supplement anti-viruses exist on end user computers. This will not only easily detect known viruses, but also acts as a knowledge that will detect advanced forms of harmful files. While a costly model requiring huge infrastructure, it can help in protecting invaluable enterprise data from security threat, and prevent immense financial damage. Our main target was to propose a machine learning framework that generally detects malware samples with the tough constraint of having a zero false positive rate. We were very close to our goal, whilst we still have a non-zero false positive rate. In order that this framework to become part of a highly competitive commercial product but a number of deterministic exception mechanisms have to be added. In our opinion, malware detection via machine learning will not replace the conventional detection methods used by anti-virus vendors, but will come as an added feature to them. Any commercial anti-virus product is subject to certain speed and memory limitations, therefore the most reliable algorithms among those presented here are the cascade one-sided perceptron. As of this moment, our framework was proven to be a effective research tool. For the near future we plan to integrate more classification algorithms to it, for instance large margin perceptrons and Support Vector Machines.

REFERENCES

[1] Bill Howe University of Washington, Coursera, Introduction to data science data_mining/ information_gain.txt.

[2] M. Christodorescu, S.Jha, Static analysis of executables to detect malicious patterns, In Proceedings of the 12th USENIX Security Symposium, Washington, pp. 105-120, 2003.

[3] Defining Malware: FAQ. http://technet.microsoft.com. Retrieved 2009-09-10.

[4] E. Konstantinou, Metamorphic virus: Analysis and detection, Technical Report RHUL-MA-2008-2, Search Security Award M.Sc. thesis, 93 pages, 2008.

[5] E. Filiol, Computer Viruses: from Theory to Applications, New York, Springer, 2005.

[6] E. Filiol, G.Jacob, M.L.Liard, Evaluation methodology and theoretical model for antiviral behavioral detection strategies, Journal in Computer Virology, pp 27-37, 2007.

[7] F-Secure Corporation. F-Secure Reports Amount of Malware Grew by 100% during 2007, Press release Retrieved 2007-12-11.

[8] History of Viruses, http://csrc.nist.gov/publications /nistir/threats/subsubsection3_3_1_1.html.

[9] http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learning.

[10] https://gerardnico.com/data_mining/information_ gain# humidity.

[11] https://github.com/surajr/Machine-Learning-approach-for Malware detection/blob/master/ malware-classification.ipynb

[12] H. Witten, E. Frank, Data mining: Practical machine learning tools with Java implementations, Morgan Kaufmann, 2005.

[13] I. Santos, Y. K. Penya, J. Devesa, P. G. Garcia, N-grams-based file signatures for malware detection, 2009.

[14] I. Yoo, Visualizing Windows executable viruses using self-organizing maps, Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, New York, pp. 82-89, 2004.

[15] Landesman, Mary, What is a Virus Signature? Retrieved 2009-06-18.

[16] M. Chandrasekaran, V. Vidyaraman, S. J. Upadhyaya, Spycon: Emulating user activities to detect evasive spyware, IEEE Computer Society, pp. 502-509, 2007.

[17] M. Stamp, S. Attaluri, S. McGhee, Profile hidden markov models and metamorphic virus detection, Journal in Computer Virology, 2008.

[18] N. Cristianini, J. Shawe-Taylor, An introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press.

 DOI : 10.18231/2454-9150.2018.1173 © 2018, IJREAM All Rights Reserved.