New malware detection framework based on N-grams and Support Vector Domain Description

Mohamed EL Boujnouni[1], Mohamed Jedra[2], and Noureddine Zahid[3]
Faculty of Sciences, Mohammed V-University, Laboratory of Conception and Systems
(Microelectronic and Informatics) Avenue Ibn Battouta B.P 1014
Rabat, Morocco
[1]med_elbouj@yahoo.fr,[2]jedra@fsr.ac.ma,[3]zahid@fsr.ac.ma

***Abstract*—Malware is a sequence of instructions that has the potential to harm any computer system or computer network. Thus detecting malware especially new ones is a critical topic in today's software security profession. Traditional signature based detection performs well against known malicious programs but can't deal with new ones where signatures are not available. Furthermore, this approach is generally regarded as ineffective against attacks like code polymorphism and metamorphism used by malware writers to obfuscate their code. To overcome this problem new techniques have been developed using data mining and machine learning. In this paper we present a new framework to detect new malicious programs, it's based on N-grams and an improved version of Support Vector Domain Description. We preprocessed and classified several hundred of computer viruses and clean programs to confirm the feasibility and the effectiveness of the proposed method.**

Keywords: Information Security, Malware detection, N-grams, Support Vector Domain Description.

I. INTRODUCTION

Malware, short of malicious software, is any type of program intended to damage a computer system, or computer network. The history of malware began with Computer Virus, a term first introduced by Cohen [19]. Malware can be defined according to McGraw and Morisett [16] as any code added, changed, or removed from a software system to intentionally cause harm or subvert the system's intended function. Other definitions was proposed by commercial anti-malware companies, which describe malware as software designed to delete, block, modify or copy data, or disrupt the performance of computers or computer networks. Based on their particular actions, malware can be categorized into groups and subgroups that can overlap:

Computer virus: A computer virus is the well-known form of malwares, it has the potential to reproduce its own code by attaching itself to other computer programs or data files in such a way that the virus code is executed when the infected file is activated. Computer viruses perform destructive tasks from relatively harmless prank messages to the destruction of valuable programs and data.

Worms: A worm is a self-replicating and fully functional malicious program. Contrary to viruses, worms often exploit local area network or Internet connection to spread without needing to be attached to an executable or document file. When a worm finds a new host, it replicates itself into that host and continues to look for others on which to replicate, the replication continues indefinitely or until a self-timing mechanism halts the process.

Trojan: A Trojan is a malicious program that appears to perform a legitimate function but covertly do harmful activities. Unlike viruses and worms, Trojans don't replicate themselves. Once activated Trojans perform one or more destructive tasks, such as stealing identity, destructing data, providing remote access, etc.

Spyware: A spyware is a form of malicious software that captures information about users or otherwise takes control of system resources without the user's knowledge or consent. Spyware can be installed on a computer through several covert means such as including as part of a virus or as the result of installing a new program.

Adware: An adware is any software application in which advertising banners such as pop-ups or Web links are displayed while a program is running. Like spyware, adware is a major nuisance for users. It is typically used to spread advertisements providing some type of financial benefit to the attacker. Adware usually does not cause permanent damage to computers. However, it can render the system inoperable if not removed.

Rootkit: A Rootkit is a particularly dangerous from of malware; it provides the attacker with control over an infected system. Rootkits are extremely difficult to detect and eradicate since they are typically installed into low level system resources. Because of this, Rootkits often go undetected by conventional anti-virus software.

Many factors favor the spread of malware such as the growth of Internet, the advent of social networks, the vulgarization of smart devices, the increasing use of storage media, etc. Malware causes many damages to computer and networks, according to [23] over 390,000 new malicious programs appear every day. Moreover a recent threat report published [12], states that the release rate of malicious code and other unwanted programs may be exceeding that of legitimate software applications.

Detecting malicious codes has become one of the prime research interests in the field of information security. The traditional way to overcame malwares is signature based virus

detection which is adopted by the majority of commercial antivirus products. This method uses signatures of known viruses to generate detection models. This technique suffers from a lot of limitations like, code obfuscation, code displacement, compression and encryption used by malware writers to overcome the use of signatures. Also, obtaining the signatures of known malicious programs means that computer systems have been already infected. Furthermore, the increasing number of known viruses implies the growth of the size of signature's database and consequently the time of checking a file for virus signatures increases. To overcome the limitations of using signatures, researchers have turned to analysis of malware behavior. The analysis can be static, dynamic, or hybrid. The first technique mostly operates on machine-level code and disassembled instructions, it analyzes the malware without executing it. The features extracted include string signature, byte-sequence N-grams, syntactic library calls, system calls, application programming interface calls, control flow, operational code frequency distribution, etc. The second technique executes the suspicious code in a protected environment (virtual machine, simulator, emulator, sandbox, etc.), and observes the behavior of malware and its effect on the environment. In recent years different malware researchers have proposed methods using data mining and machine learning for detecting new malicious codes. They use data mining to analyze and extract patterns from programs, and machine learning methods to classify and find correlations between the extracted patterns. The methods based on data mining and machine learning have shown good results compared to other approaches.

In this paper, we present a new framework for malware detection. It's based on N-grams method to extract features from executable files, and a new improved version of Support Vector Domain Description to classify the extracted features and to detect new unknown malwares. This paper is organized as follows, the second section provides some background information on malware detection using N-grams and machine learning methods, the third section presents the proposed framework, the fourth section provides an experimental evaluation of the proposed technique, the last section concludes briefly the paper.

II. RELATED WORKS

Several approaches for static malware detection have been studied, we focus in using N-grams method combined with machine learning techniques. The method N-grams explores the structure of a program using bytes, characters, or text strings, it was used for malware analysis by Kephart and Arnold [9] in 1994, but their work lacks experimental results. Santos et al [15] extracted N-grams profile through operational code sequences from a large database of programs, and formed signatures that are classified with machine learning techniques. Kolter and Maloof [1, 10] represented malware though N-grams as features and selected top 500 N-grams according to information gain measure [10]. Many learners like term frequency inverse document frequency (TF-IDF), K-Nearest Neighbor, naïve Bayes, Support Vector Machines, decision trees, and their boosting versions were applied to generate rules for classifying new binary executables. Assaleh et al [4] proposed an N-grams-based signature method to detect computer viruses. Using a set of software they generated N-grams profiles for each class of software and for classifying any unknown instance into benign or malicious code, they employed a k-NN algorithm with $k=1$. Schultz et al [2] extracted features with N-grams from a set of malicious programs using three approaches (i.e the list of dynamic link library calls, strings sequences and byte sequences), the extracted features were trained using Ripper - a rule based learner -, naïve Bayes, and Multi-naïve Bayes. Henchiri and Japkowicz [8] collected features with N-grams and selected them hierarchically, the result were classified using decision trees, naïve Bayes, and Support Vector Machine. Zhang et al [11] used N-grams to extract features and selected the most relevant of them by calculating their information gain, the results are provided as input for probabilistic neural network classifier. O. Sornil and C. Liangboonprakong [14] created features from executable files in four sequential steps, N-grams extraction, sequential pattern extraction, pattern statistics calculation, and feature reduction. Three learning algorithms, namely C4.5, multilayer perceptron, and Support Vector Machine were used to perform the classification. Elovici et al [13] represented executable files with N-grams, they reduced features using fisher score, and trained the result with Bayesian networks, artificial neural networks, and decision trees. P. O'Kane et al [24] used SVM to determine the ability of N-grams to correctly predict the presence of malware. They found that an N-gram size N=3 and N=4 present the best results. G. Canfora et al [25] investigated the effectiveness of Opcode N-grams for Detection of Multi Family Android Malware. Their experiments on large dataset (11120 applications) showed that a perfect detection rate was achieved for more than one malware family.

III. THE PROPOSED APPROACH FOR DETECTING MALWARE

As said before our aim is to build a new framework to detect malwares using static analysis. To do that we started by extracting N-grams from a set of benign and malicious files, based on their information gain measure we identified the most relevant N-grams, then an improved version of SVDD, namely Support Vector Domain Description with Small Sphere and Parametric Volume SSPV-SVDD [21] was used to classify the extracted N-grams and to detect new unknown malwares. In what follows we present in details the proposed approach.

A. The method N-grams

The concept of N-grams was first discussed in 1951 by Shannon [20]; N-grams has been used in many areas, such as probability communication theory, computational linguistics, biological sequence analysis, data compression, etc. The N-grams approach has been extensively used in malware detection [4, 14, 15, 18, 22]. The N-grams signature of a string is a set of all its substrings with a length N. In the case of a binary executable, the N-grams signature is usually computed on the string of its raw bytes, or disassembled instructions. To generate the N-grams vector, a window of N character is moved through the text, sliding forward one character at a time. At each position of the window, the sequence of characters in the window is recorded. For example, the string "malware", can be segmented into several 4-grams: "malw", "alwa",

"lwar", "ware". In some schemes, the window may be slid more than one character after each N-grams is recorded.

Applying N-grams to an executable file starts from generating a feature set, for example the hexadecimal byte sequence, and then extracting distinct N-grams and their frequency of appearance.

TABLE I. THE FIRST 4-GRAMS EXTRACTED FROM A MALICIOUS FILE

4-grams	Associated frequency
40008BC0	89
008BC0FF	82
8BC0FF25	82
FFFFFFFF	78
00FFFFFF	69
9140008B	59
000000E8	57
01000000	56

As the set of all N-grams is very large, it's recommended to use the most relevant of them, based on their information gain. The information gain of an N-gram is given in [22]:

$$IG(N_g) = \sum_{V_{Ng}\in\{0,1\}} \sum_{C\in\{C_i\}} P(V_{Ng}, C)\log\frac{P(V_{Ng}, C)}{P(V_{Ng})P(C)} \quad (1)$$

Where C is one of two classes benign or virus, V_{Ng} is the value of the N-gram, $V_{Ng} = 1$ if Ng occurs in the program and = 0 otherwise. $P(V_{Ng}, C)$ is the proportion of programs in C in which that Ng takes on value V_{Ng}. $P(V_{Ng})$ is the proportion of programs in entire training set such that Ng takes the value V_{Ng}. $P(C)$ is the proportion of the training data belonging to the class C.

B. An improved Support Vector Domain Description

Support Vector Domain Description (SVDD) was developed by Tax and Duin [5, 6, 7] to solve one-class classification problems. Inspired by the Support Vector Machine learning theory, SVDD aims to find the smallest hypersphere containing the most of the target data. Due to the use of kernel tricks, SVDD can map the training set into a higher dimensional feature space and then find a minimal hypersphere that can enclose all or most of the mapped target data in this feature space. An improvement of SVDD called SSPV-SVDD was proposed [21], SSPV-SVDD aims to search the minimal enclosing hypersphere, following the value of a parameter called p, which controls the volume of the hypersphere and plays a compromise between the acceptance of negative data and the rejection of target data. In what follows we present a summarized description of SSPV-SVDD.

Suppose we are given training set of instance-label pairs $(x_i, y_i), i = 1,..,N$ with $x_i \in R^d$, the value y_i is +1 for the target samples, and -1 for the negative ones. We aim to find the smallest hypersphere of radius R that encloses only the positive samples, which is described by the following constraints:

$$\|x_j - a\|^2 \le R^2 - p.y_j + \varepsilon_j \ \ \forall j \ with \ y_j = +1 \quad (2)$$
$$\|x_j - a\|^2 \ge R^2 - p.y_j - \varepsilon_j \ \ \forall j \ with \ y_j = -1 \quad (3)$$

The ε_j are slack variables that measure the amount of violation of the constraints, p is a positive real number. To solve this problem the Lagrangian was introduced:

$$L(R, \varepsilon, a) = R^2 + C\sum_{i=1}^{N}\varepsilon_i - \sum_{i=1}^{N}\alpha_i y_i(R^2 - \|x_i - a\|^2 - p.y_i) - \sum_{i=1}^{N}\varepsilon_i\mu_i \quad (4)$$

Where α_i and μ_i are Lagrange multipliers, Setting the partial derivatives of L with respect to R, a, ε_i to zero gives the following constraints:

$$\frac{\partial L}{\partial R} = 0 \Rightarrow \sum_{i=1}^{N}\alpha_i y_i = 1 \quad (5)$$
$$\frac{\partial L}{\partial a} = 0 \Rightarrow a = \sum_{i=1}^{N}\alpha_i x_i y_i \quad (6)$$
$$\frac{\partial L}{\partial \varepsilon_i} = 0 \Rightarrow \alpha_i = C - \mu_i \quad (7)$$

The dual optimization problem becomes:
Maximize:

$$L(\alpha) = -\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i x_j + \sum_{i=1}^{N}\alpha_i(y_i x_i x_i + p) \quad (8)$$

Subject to

$$0 \le \alpha_i \le C \quad et \quad \sum_{i=1}^{N}\alpha_i y_i = 1 \quad (9)$$

For multi-class classification problems, multiple optimized hyperspheres which describe each class were constructed separately. To classify a new, unknown sample z, we just investigate whether it's inside the hypersphere with the center a_k and the radius R_k associated to the class k and constructed during the training [3][5][6][7] . The decision function is given by the equation (10), if its value is positive for the k^{th} class and negative for the others, then we conclude that z belongs to the class k.

$$f(z) = sgn(R_k^2 - \|z - a_k\|^2) \quad (10)$$

The radius R_k associated to the k^{th} class, can be defined as follows:

$$R_k^2 = \|x_s - a_k\|^2 + y_s.p \quad (11)$$
$$R_k^2 = x_s.x_s - 2\sum_{j=1}^{N}x_j x_s y_j\alpha_j + \sum_{i=1}^{N}\sum_{j=1}^{N}x_i x_j y_i y_j\alpha_i\alpha_j + y_s.p \quad (12)$$

Where x_s is a training point that belongs to the set of Support Vectors having $0 < \alpha_s < C$, and y_s is the label of x_s.

IV. EXPERIMENTAL RESULTS

A. Datasets and experimental setting

The experimental dataset consists of a total of 1258 Windows files that are: 600 benign executables collected from different versions of operating systems (windows XP and Windows 7), and 658 malicious programs downloaded from VX Heaven website [17]. The detailed distribution of the proposed dataset is shown in figure 1.The evaluation of our method was performed through three experiments, which aim to classify the dataset previously described using the method N-grams and the classifiers SVDD and SSPV-SVDD. The first experiment was performed by one family of virus named Dos.Vienna against benign files, while the second complicates the description by using the combination of three unbalanced

families of virus namely Dos.BadBoy, Dos.Ash and Dos.BW against benign files. The last test evaluates the discrimination between virus families using two samples of them (Dos.Burger and Dos.Austalian). The evaluation of our approach was done in two steps:

- **Training**: We extract the set of distinct N-grams, and their frequency of appearance from each executable file from the training dataset, then we combine N-grams of all files (duplicates to be eliminate), to construct a common base, in which all N-grams will be projected. After, we calculate the information gain (IG) of each N-grams, then we sort them in descending order of IG, we form a new reduced base by selecting the first k top N-grams. At the end of this step each executable file (malware and benign) is represented by its signature (k N-grams). Using the set of signatures, we run SVDD and SSPV-SVDD with different values of σ and we calculate the recognition rate.
- **Evaluating**: In this step we evaluate the membership of each new, unknown file from the testing dataset, we extract its distinct N-grams and their frequency of appearance, we form its signature by projecting the set of its N-grams in the reduced base (built in training step), we evaluate its membership using the decision function of SVDD and SSPV-SVDD.

For all experiments we use the Gaussian kernel since it's usually used in pattern recognition problems and it achieves good results, the parameters C and k take the values 100 and 50 respectively. The procedure is repeated ten times, then the corresponding results are averaged, and the standard deviations are calculated. Figure 2 illustrates the proposed framework.

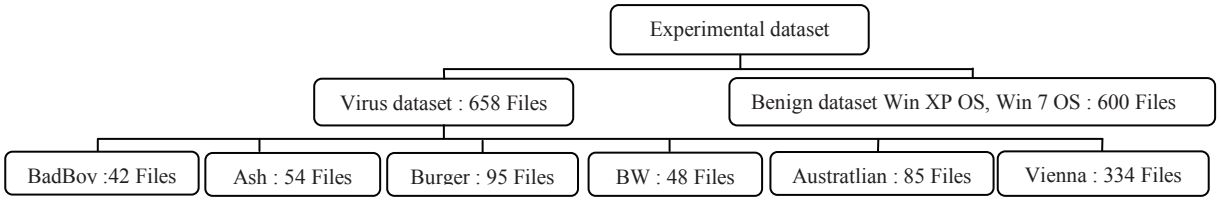

Figure 1. The distribution of the executable files

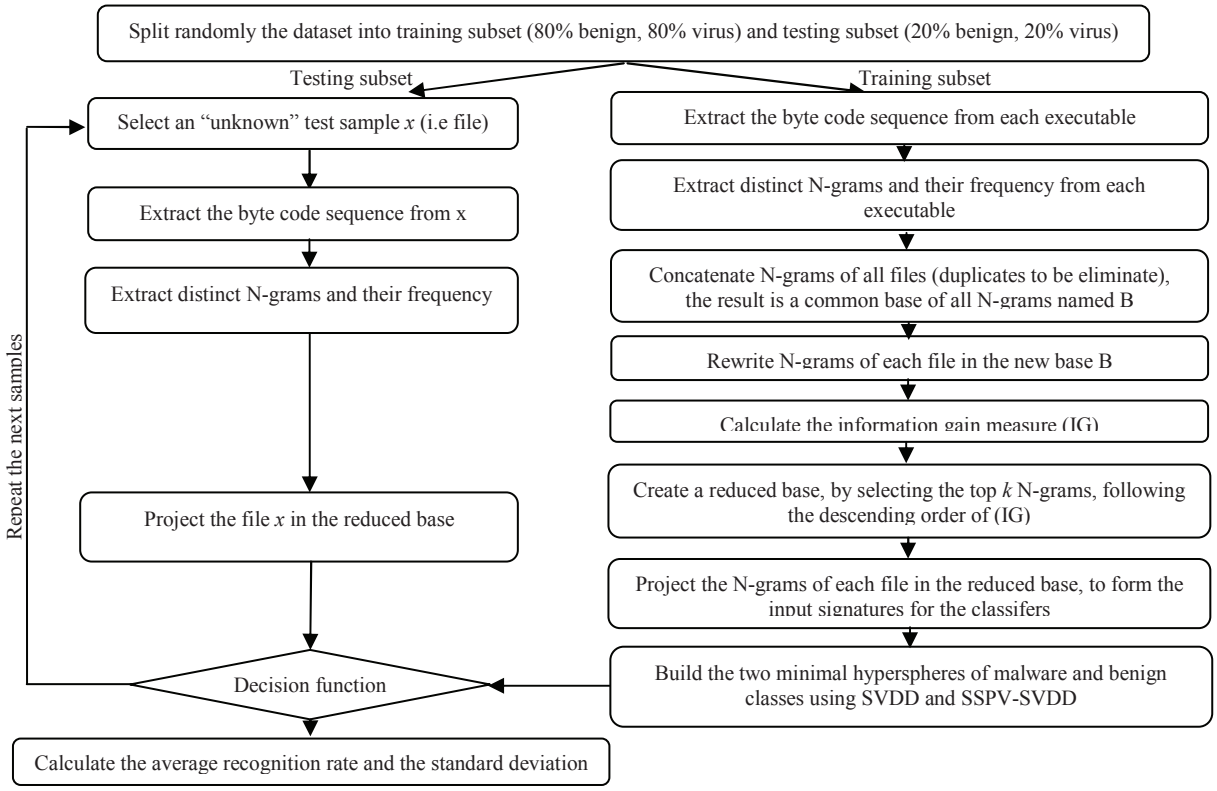

Figure 2. Architecture of the proposed malware detection framework

B. Numerical Results

The first experiment (Figure 3) was performed by a single family of viruses namely Dos.Vienna, it shows the average recognition rate and the standard deviation of SVDD and SSPV-SVDD, using different values of σ. The figure is divided into two parts:

- The training dataset: It can be seen that, when we run SVDD, the average recognition rate reaches a maximum of 74% and expresses larger variance, by contrary to SSPV-SVDD which is stable and achieves approximately 100% whatever the value of σ.
- The testing dataset: It can be observed that the average recognition rate using SSPV-SVDD outperforms SVDD (maximum of 97% and 58 % respectively), also the standard deviation of SSPV-SVDD is lower than SVDD.

The second experiment (Figure 4) has the same objective as the first one, but it complicates the task of SVDD and SSPV-SVDD, since it constructs a new heterogenic set of virus, based on the combination of three unbalanced families of viruses: Dos.BadBoy, Dos.Ash and Dos.BW. The figure is divided into two parts:

- The training dataset: It can be seen that, on average 100% of the training executables are correctly classified using SSPV-SVDD, but when switching to SVDD the results decrease drastically (maximum of 70%) and the variance become more important.
- The testing dataset: It can be observed that, approximately 94% of new executables are correctly assigned to the right class, with low variance when using SSPV-SVDD, in contrast to SVDD which shows poor results (maximum of 44%) with large variance.

The third experiment (Figure 5) aims to evaluate the ability of SVDD and SSPV-SVDD to discriminate between different families of viruses. For this purpose SVDD and SSPV-SVDD were trained by Dos.Burger and Dos.Austalian. The figure can be divided into two parts :

- The training dataset: It shows that the best recognition rate with a low variance were achieved using SSPV-SVDD (maximum of 95%) compared to SVDD (maximum of 85%).
- The testing dataset: It shows that the classifier SSPV-SVDD gives better recognition rate (maximum of 90%) than SVDD (maximum of 82%).

All experiments demonstrate that the best classification accuracy was given by the combination of N-grams and SSPV-SVDD.

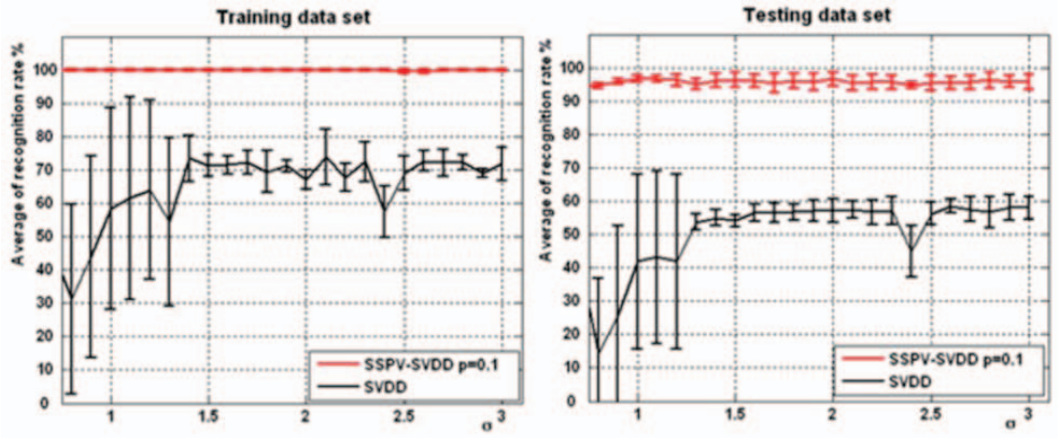

Figure 3. Comparison between the classification accuracy of SVDD and SSPV-SVDD using one type of viruses (Dos.Vienna) against benign files

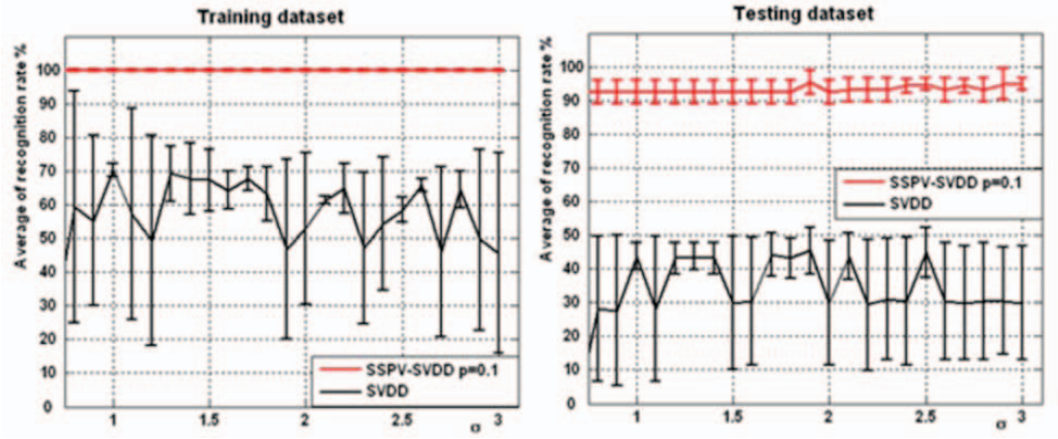

Figure 4. Comparison between the classification accuracy of SVDD and SSPV-SVDD using three types of viruses (Dos.BadBoy, Dos.Ash and Dos.BW) against benign files

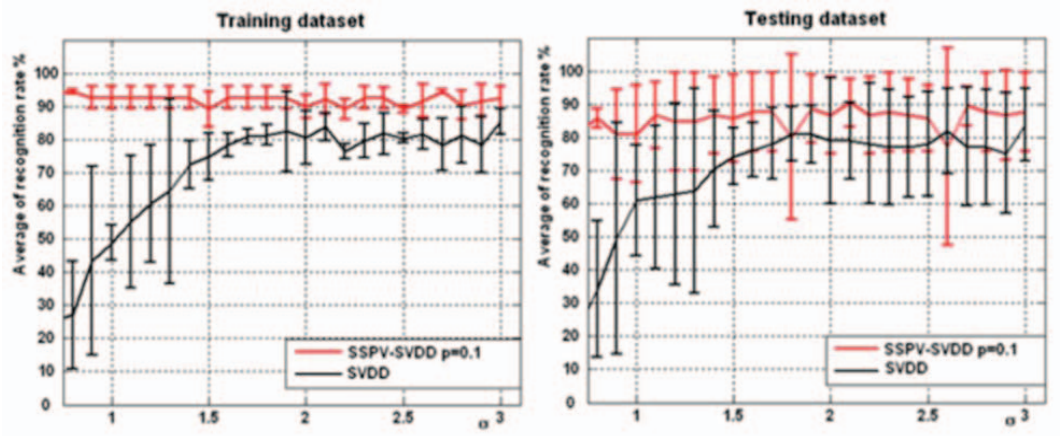

Figure 5. Comparison between the classification accuracy of SVDD and SSPV-SVDD using two different types of viruses (Dos.Burger Vs Dos.Austalian)

V. CONCLUSIONS

In this paper, we presented a new method to detect unknown malwares. It's based on two techniques namely N-grams and an improved version of SVDD named SSPV-SVDD. The first technique was used to extract features from executable files, while the second was applied to classify the extracted features, and to detect new unknown malicious codes. The experimental results, using a set of 658 malicious programs belonging to six families of computer viruses against 600 benign programs show that N-grams with SSPV-SVDD can achieve a maximum detection accuracy of 97% with one family of viruses, 94% with three families of viruses and 90% of accuracy to discriminate between different families of viruses. Based on these results, we can conclude that our method can detect successfully new unknown malicious programs with good accuracy.

REFERENCE

[1] J. Z. Kolter and M. A. Maloof. "Learning to detect malicious executables in the wild". In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 470-478, New York, USA, 2004.

[2] M.G. Schultz, E. Eskin, E. Zadok and S.J. Stolfo, "Data Mining Methods for Detection of New Malicious Executables". In Proceedings of the IEEE Symposium on Security and Privacy, pp. 38-49, Oakland, CA, 2001.

[3] A. Tavakkoli, M. Nicolescu, G. Bebis, and M. Nicolescu. "A support vector data description approach for background modeling in videos with quasi-stationary backgrounds". International journal on artificial intelligence tools, Vol. 17, N°. 4, pp. 635-658, 2008.

[4] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan. "N-grams-based detection of new malicious code". In Proceedings of the 28th International Computer Software and Applications Conference COMPSAC, Vol. 2, pp. 41-42, Hong Kong, China, 2004.

[5] D.M.J. Tax, and R.P.W. Duin. "Data Domain Description Using Support Vectors". In Proceedings- European Symposium on Artificial Neural Networks, pp 251-256, Bruges (Belgium), 1999.

[6] D.M.J. Tax, and R.P.W. Duin . "Support Vector Data Description". Machine Learning. Vol. 54, pp 45-66, 2004.

[7] D.M.J. Tax, and R.P.W. Duin. "Support vector domain description". Pattern Recognition Letters, Vol. 20, N°. 11-13, pp. 1191–1199, 1999.

[8] O. Henchiri and N. Japkowicz. "A feature selection and evaluation scheme for computer virus detection". In Proceedings of the 6th International Conference on Data Mining, pp. 891-895, Hong Kong , 2006.

[9] J. O. Kephart and W. C. Arnold, "Automatic extraction of computer virus signatures". In Proceedings of the 4th International Virus Bulletin Conference, pp. 179-194. Virus Bulletin Ltd, 1994.

[10] J.Z. Kolter and M.A. Maloof. "Learning to detect and classify malicious executables in the wild". The Journal of Machine Learning Research, Vol. 7, pp. 2721-2744, 2006.

[11] B. Zhang, J. Yin, J. Hao, D. Zhang, and S. Wang. "Malicious codes detection based on ensemble learning". Autonomic and Trusted Computing, pp. 468-477, 2007.

[12] "Symantec Internet Security Threat Report:Trends for July–December 2007" Volume 13. pp. 29, Symantec Corporation. April 2008.

[13] Y. Elovici, A. Shabtai, R. Moskovitch, G. Tahan, and C. Glezer. "Applying machine learning techniques for detection of malicious code in network traffic". KI 2007: Advances in Artificial Intelligence, Vol. 4667, pp. 44-50, 2007.

[14] O. Sornil and C. Liangboonprakong, "Malware Classification Using N-grams Sequential Pattern Features". International Journal of Information Processing and Management(IJIPM), Vol. 4, N°.5, pp 59-67, July 2013

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

[16] G. McGraw and G. Morrisett. "Attacking malicious code : A report to the infosec research council" . IEEE Software, Vol. 17, N°.5, pp: 33-44, 2000.

[17] VX Heavens, http://www.vxheaven.org

[18] J. Jang, D. Brumley, and S. Venkataraman. "Bitshred: feature hashing malware for scalable triage and semantic analysis". In Proceeding of the 18th ACM Conference on computer and communication security, 2011.

[19] Fred Cohen. "Computer Viruses". PhD thesis, University of Southern California, 1985.

[20] C. E. Shannon, "Prediction and entropy of printed English," Bell System Technical Journal,Vol. 30,N°. 1, pp. 50- 64, 1951.

[21] M. EL Boujnouni, M. Jedra, and N. Zahid, "A Small Sphere and Parametric Volume for Support Vector Domain Description", Journal of Theoretical and Applied Information Technology. Vol. 46 N°.1, pp. 471-478, 2012.

[22] D. Krishna Sandeep Reddy, K. Arun Pujari, "N-gram analysis for computer virus detection", Journal in Computer Virology, Vol. 2, N°. 3, pp. 231-239, 2006.

[23] AVTEST The Independent IT-Security Institute : Http://www.av-test.org/en/statistics/malware/

[24] P. O'Kane, S. Sezer, K.McLaughlin, "N-gram density based malware detection," In World Symposium on Computer Applications & Research (WSCAR), pp 1-6, Sousse, Tunisia, 18-20 Jan 2014.

[25] G. Canfora, A. De Lorenzo, E. Medvet, F. Mercaldo, "Effectiveness of Opcode ngrams for Detection of Multi Family Android Malware", In proceedings of the 10th International Conference on Availability, Reliability and Security (ARES), Toulouse, France, 24-27 Aug. 2015.