SATTVA: SpArsiTy inspired classificaTion of malware VAriants

Lakshmanan Nataraj
Dept. of ECE
University of California, Santa Barbara, USA
lakshmanan_nataraj
@ece.ucsb.edu

S. Karthikeyan
Dept. of ECE
University of California, Santa Barbara, USA
karthikeyan
@ece.ucsb.edu

B.S. Manjunath
Dept. of ECE
University of California, Santa Barbara, USA
manj
@ece.ucsb.edu

ABSTRACT

There is an alarming increase in the amount of malware that is generated today. However, several studies have shown that most of these new malware are just variants of existing ones. Fast detection of these variants plays an effective role in thwarting new attacks. In this paper, we propose a novel approach to detect malware variants using a sparse representation framework. Exploiting the fact that most malware variants have small differences in their structure, we model a new/unknown malware sample as a sparse linear combination of other malware in the training set. The class with the least residual error is assigned to the unknown malware. Experiments on two standard malware datasets, Malheur dataset and Malimg dataset, show that our method outperforms current state of the art approaches and achieves a classification accuracy of 98.55% and 92.83% respectively. Further, by using a confidence measure to reject outliers, we obtain 100% accuracy on both datasets, at the expense of throwing away a small percentage of outliers. Finally, we evaluate our technique on two large scale malware datasets: Offensive Computing dataset (2,124 classes, 42,480 malware) and Anubis dataset (209 classes, 36,784 samples). On both datasets our method obtained an average classification accuracy of 77%, thus making it applicable to real world malware classification.

Categories and Subject Descriptors

K.6.5 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Security and Protection—*Invasive software*; I.5.4 [**PATTERN RECOGNITION**]: Applications—*Signal Processing*

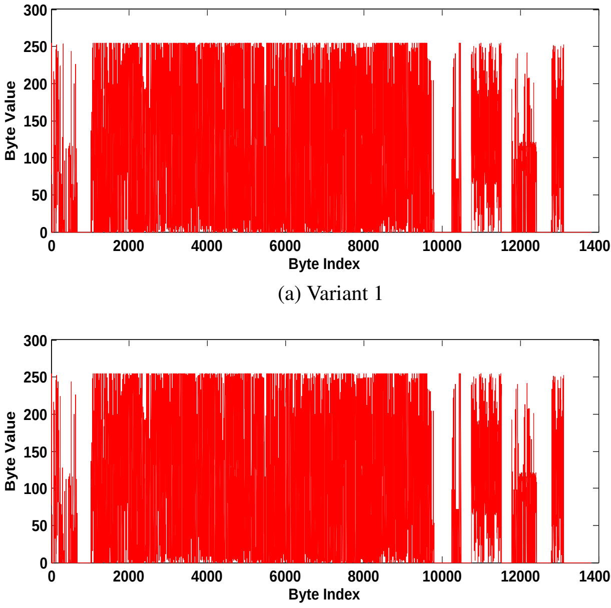
Keywords

Malware Variant Classification, Sparsity based classification, Random Projections, Compressed Sensing

1. INTRODUCTION

Antivirus (AV) software vendor Kaspersky recently reported that they process on average 315,000 samples per day [1]. The main reason for such a deluge is *malware mutation: the process of creating new malware variants from existing ones.* Variants are created either by making changes to the malware code or by using executable packers. In the former case, simple mutation occurs by changing small parts of the code. In the latter case, a more complex mutation occurs by encrypting (usually with different keys) the main body of the code and appending a decryption routine, which during runtime decrypts the encrypted payload. The new variants perform the same function as the original malware but their attributes would be so different that AV software, which use traditional signature based detection, would not work on them. Based on their function, these variants are classified into different *malware families*. Identifying the malware family of an unknown malware can play an important role in understanding and thwarting new attacks.

Although mutation techniques create a large number of malware variants, these variants have very small changes in the overall malware structure. Fig. 1 shows one such example of two malware variants of the recently exposed Regin malware [3], which has been



(a) Variant 1

(b) Variant 2

Figure 1: Byte plots of the recently exposed Regin malware variants [3]. Variant (b) is created by making small change to Variant (a). They differ only in 7 bytes out of 13,284 bytes (0.0527 %).

IH&MMSec'15, June 17–19, 2015, Portland, Oregon, USA.
Copyright © 2015 ACM 978-1-4503-3587-4/15/06 ...$15.00.
DOI: http://dx.doi.org/10.1145/2756601.2756616.

described as one of the most sophisticated malware discovered in recent times, and termed on par with Stuxnet, Flame and other advanced malware. In Fig. 1, variants are represented as byte plots where every byte is represented as a number. Despite their sophisticated design, the variants only differ in a few bytes (0.0527 %). Reports further observe that variants of Regin malware were used for diverse tasks such as cyber-espionage and secret surveillance against countries, companies and individuals. Although this example shows a case of simple mutation, this phenomenon is also true for variants created using executable packers, which are more common nowadays.

In this paper, we explore Sparse Representation based Classification (SRC) methods to classify malware variants into families. Such methods have been previously applied to problems where samples belonging to a class have small variations in them, for example, face recognition [31], iris recognition [27], background subtraction [6], and tracking [21]. We model a malware variant belonging to a particular malware family as a sparse linear combination of variants from that family using Random Projections. Since variants of a family have small changes in the overall structure and differ from variants of other families, projections of malware in lower dimensions preserve this “similarity”.

The rest of the paper is organized as follows. The related works in malware classification are briefed in Sec. 2. Sec. 3 details the formulation of the sparse representation based classification framework. Sec. 4 details the experiments on various datasets. The limitations and conclusion are discussed in Sec. 5.

2. RELATED WORK

Typical malware features used for malware classification can be broadly grouped into either *static features* or *dynamic features* As the name suggests, static features are extracted from the malware without executing it. Dynamic features, on the other hand, are extracted by executing the code, usually in a virtual environment, and then studying their behavioral characteristics such as system calls trace or network behavior. We will focus more on static analysis which our proposed approach comes under. For more on dynamic analysis based methods, the readers are referred to [5, 28, 10, 8].

The most common static analysis method is control flow graph analysis [18, 11, 13]. After disassembling the code, the control flow of the malware is obtained and graphs are constructed to uniquely characterize the malware. However, these methods do not work well on packed malware since the control flow of a packed malware reveals only the unpacking routine. In contrast, our proposed method does not require any code analysis, unpacking or execution of malware.

Other Static features are based on n-grams [4, 16, 12, 13, 26], n-perms [14, 19], hashes [17, 30] and image similarity [22, 24, 23, 15]. The first two compute n-grams or n-perms on the binaries to characterize the malware. Among hash based methods, *ssdeep* [17] is a common technique to compute context triggered piecewise hashes on raw binaries. *Pehash* [30], however, uses the Portable Executable (PE) file structure to compute a hash. Image similarity based methods [22, 24] convert a malware binary to a digital image and apply image processing based techniques to compute features. These features have been used for malware classification [22], detection [15] and retrieval [23]. In contrast to these methods, we compute random projections on malware represented as numerical vectors. This results in compact features for malware classification. Although random projections have been previously used in [10, 8], these methods require dynamic analysis which is time consuming.

3. MALWARE CLASSIFICATION BASED ON SPARSE REPRESENTATIONS

3.1 Approach

Given a dataset of N labeled malware belonging to L different malware families with P malware per family, the task is to identify the family of an unknown malware $\mathbf{u}$. Similar to [22], we represent a malware as a numerical vector $\mathbf{x}$ of range $[0, 255]$, where every entry of $\mathbf{x}$ is a byte value of the malware. However unlike [22], we do not convert this vector to an image matrix. Since each malware sample can have a different code-length, we normalize all vectors to a maximum length (M) by zero-padding.

The entire dataset can now be represented as an $M \times N$ matrix $\mathbf{A}$, where every column represents a malware. Further, for every family k ($k = 1, 2, ..., L$), we define an $M \times P$ matrix $\mathbf{A}_k = [\mathbf{x}_{k1}, \mathbf{x}_{k2}, ...\mathbf{x}_{kP}]$ where $\mathbf{x}_{k\{.\}}$ represents a malware sample belonging to family k. Now, $\mathbf{A}$ can be expressed as a concatenation of block-matrices $\mathbf{A}_k$:

$$\mathbf{A} = [\mathbf{A}_1 \mathbf{A}_2 .. \mathbf{A}_L] \in \mathbb{R}^{M \times N} \tag{1}$$

Let $\mathbf{u} \in \mathbb{R}^M$ be an unknown malware whose family is to be determined, with the assumption that $\mathbf{u}$ belongs to one of the families in the dataset. Since variants in a family have small differences, they will all be in the same linear span [1]. Then, following [31], we represent $\mathbf{u}$ as a sparse linear combination of the training samples as:

$$\mathbf{u} = \sum_{i=1}^{L} \sum_{j=1}^{P} \alpha_{ij} \mathbf{x}_{ij} = \mathbf{A}\alpha \tag{2}$$

where $\alpha = [\alpha_{1,1}, ..., \alpha_{L,P}]^T$ represents the $N \times 1$ sparse coefficient vector ($N = LP$). α will have non-zero values only for samples that are from the same family as $\mathbf{u}$. The sparsest solution to (2) can be obtained using Basis Pursuit [27] by solving the following l_1-norm minimization problem:

$$\hat{\alpha} = \underset{\alpha' \in \mathbb{R}^N}{\arg\min} \, \|\alpha'\|_1 \text{ subject to } \mathbf{u} = \mathbf{A}\alpha' \tag{3}$$

where $\|.\|_1$ is the l_1 norm.

Estimating the family of $\mathbf{u}$ is done by computing residuals for every family in the training set and then selecting the family that has minimum residue. Let $\mathbf{\Pi}_k$ be the characteristic function that selects the coefficients from $\hat{\alpha}$ that are only associated with family k. Then the residual function r_k can be expressed as:

$$r_k(\mathbf{u}) = \|\mathbf{u} - \mathbf{A}\mathbf{\Pi}_k(\hat{\alpha})\|_2 \tag{4}$$

$$c = \underset{k}{\arg\min} \; r_k(\mathbf{u}) \tag{5}$$

where c is the index of the estimated family associated with $\mathbf{u}$.

3.2 Random Projections

When a malware binary is represented as a numerical vector by considering every byte, the dimensions of that vector can be very high. For example, a 1 MB malware has around 1 Million bytes and this could make the calculations computationally expensive. Hence, we project the vectors to lower dimensions using Random Projections (RP). This also removes dependency on any particular feature extraction method. Previous works have demonstrated that SRC is effective in lower-dimensional random projections as well,

[1] Linear span means that any linear combination of a vector will be in the same subspace

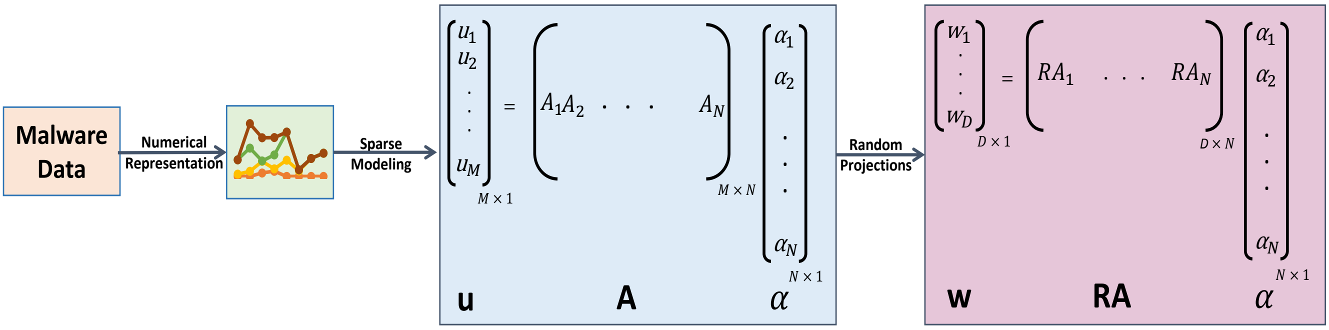


Figure 2: Overall Approach: Malware samples are represented as numerical vectors, projected to lower dimensions and then modeled using the Sparse Representation based Classification (SRC) framework

see [9, 31, 27]. Let $\mathbf{R} \in \mathbb{R}^{D\times M}$ be the matrix that projects $\mathbf{u}$ from signal space M to $\mathbf{w}$ of lower dimensional space D ($D << M$):

$$\mathbf{w} = \mathbf{R}\mathbf{u} = \mathbf{R}\mathbf{A}\alpha \tag{6}$$

The entries of $\mathbf{R}$ are drawn from a zero mean normal distribution. The above system of equations is underdetermined and sparse solutions can be obtained by reduced l_1-norm minimization:

$$\hat{\alpha} = \underset{\alpha' \in \mathbb{R}^N}{\arg\min} \; \|\alpha'\|_1 \text{ subject to } \mathbf{w} = \mathbf{R}\mathbf{A}\alpha' \tag{7}$$

The overall approach is shown in Fig.2.

3.3 Modeling Variants

When a new variant is created from existing an malware by making small changes, both variants share some common parts. The new variant is modelled as:

$$\mathbf{u}' = \mathbf{u} + \mathbf{e_u} = \mathbf{A}\alpha + \mathbf{e_u} \tag{8}$$

where $\mathbf{u}'$ is the corrupted vector representing the new variant and $\mathbf{e_u}$ is the error vector. This can be reduced to matrix form using block matrices:

$$\mathbf{u}' = [\mathbf{A}, \mathbf{I}_M] \begin{bmatrix} \alpha \\ \mathbf{e_u} \end{bmatrix} = \mathbf{B_u}\,\mathbf{s_u} \tag{9}$$

where $\mathbf{B_u} = [\mathbf{A}, \mathbf{I}_M]$ is a $M \times (N + M)$ matrix and $\mathbf{I}_M$ is an $M \times M$ Identity matrix and $\mathbf{s_u} = [\alpha, \mathbf{e_u}]^T$. This ensures that the system of equations (9) is always underdetermined and sparse solutions can be obtained. In lower dimensions, this reduces to:

$$\hat{\alpha} = \underset{\alpha' \in \mathbb{R}^N}{\arg\min} \; \|\alpha'\|_1 \text{ subject to } \mathbf{w}' = \mathbf{B_w}\,\mathbf{s_w}$$

$$r_k(\mathbf{w}') = \|\mathbf{w}' - \mathbf{B_w}\,\mathbf{s_w}\mathbf{\Pi}_k(\hat{\alpha})\|_2$$

$$c = \underset{k}{\arg\min} \; r_k(\mathbf{w}') \tag{10}$$

where $\mathbf{w}' = \mathbf{w} + \mathbf{e_w}$, $\mathbf{B_w} = [\mathbf{R}\mathbf{A}\alpha, \mathbf{I}_D]$ is a $D \times (N + D)$ matrix, $\mathbf{I}_D$ is a $D \times D$ Identity matrix and $\mathbf{s_w} = [\alpha, \mathbf{e_w}]^T$ We will use (10) to identify the malware family of an unknown test sample.

4. EXPERIMENTS

We test our technique on two public malware datasets: Malimg Dataset [22] and Malheur Dataset [28]. On both datasets, we select equal number of samples to reduce any bias towards a particular family [20]. The data is converted to numerical form and represented as a matrix as defined in (1) and then projected to lower dimensions using Random Projections (RP). For comparison, we use GIST features [25], which have been previously applied for malware classification [22]. We use the SRC framework (10) to identify the malware family of a test sample and compare with Nearest Neighbors (NN) classification that was previously used in [22]. We vary the dimensions from $\{48, 96, 192, 256, 384, 512\}$, which are consistent for both RP and GIST. In our experiments, we chose 80% of a dataset for training and 20% for testing.

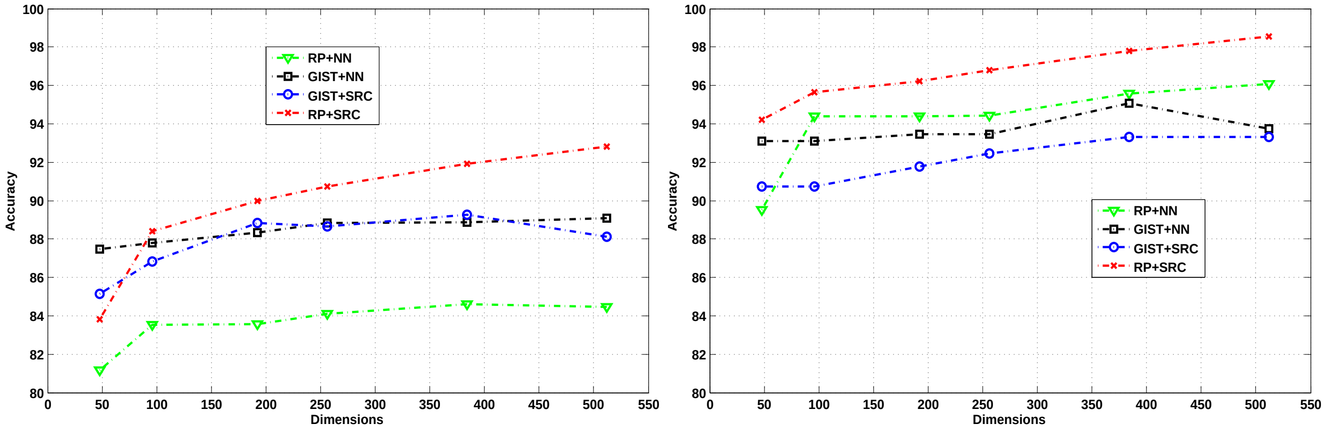
4.1 Classification

4.1.1 Results on Malimg Dataset

The Malimg dataset contains 25 malware families with 9,342 samples, which we obtained from the authors of [22]. The dataset has a mixture of both packed and unpacked malware and the number of samples per family varies from 80 to 2,949. In our experiments, we select 80 samples per family (the minimum number present in all families). The size of the largest malware (M) was 840,960 bytes and all samples were zero padded to this size. The results are shown in Fig. 3a. First, we see that the classification accuracy increases as the dimensionality increased from 48 to 512. Beyond 512, there was no significant change in accuracy for both GIST and RP. The best accuracy of **92.83%** was obtained for RP with SRC as the classifier. At the same dimension, the lowest accuracy was for RP with NN as the classifier (84.45%). The accuracies for GIST for both classifiers were almost the same, in the middle range (88-89%).

4.1.2 Results on Malheur Dataset

The Malheur dataset consists of 3,131 malware binaries from 24 malware families, which we obtained from the authors of [28]. The malware binaries were labeled such that a majority amongst six different antivirus products shared similar labels. The number of samples per family varied between 20 and 300. We chose 20 samples from all families in our experiments. For this dataset, the value of M was 3,364,864. The classification results are shown in Fig. 3b. Here too, the best accuracy of **98.55%** was obtained for RP at 512 dimensions with SRC as classifier. However, unlike the Malimg dataset, RP with NN as classifier also had a high accuracy of 96.06%. This shows that the random projections of the variants in Malheur dataset are closely packed in lower dimensions. On the other hand, the accuracies for GIST features were around 93% for both classifiers.



(a) Malimg Dataset

(b) Malheur Dataset

Figure 3: Experimental Results on (a) Malimg Dataset and (b) Malheur Dataset with features using Random Projections (RP) and GIST, and classification algorithm using Sparse Representation based Classification (SRC) and Nearest Neighbor (NN).

4.2 Comparison with Other Features

We compare our proposed approach with other relevant malware similarity features: *ssdeep* [17], GIST [22] and n-grams [16]. For n-grams, we chose $n = 2$ and computed a 2^{16} dimensional feature vector. The results are shown in Tab. 1. For both datasets, our proposed approach outperformed *ssdeep*, GIST and n-grams based features.

Table 1: Comparison of Classification Accuracies

Dataset	*ssdeep*	GIST	n-grams	RP
Malimg Dataset	67.63	89.08	91.75	**92.83**
Malheur Dataset	81.6	94.21	94.26	**98.55**

4.3 Rejecting Outliers

In order to reject test samples that do not belong to any family in a dataset, the Sparsity Coefficient Index (SCI) of a coefficient vector $\alpha \in \mathbb{R}^N$ is defined as:

$$SCI(\alpha) = \frac{\frac{L.max\|\mathbf{\Pi}_i(\alpha)\|_1}{\|\alpha\|_1} - 1}{L - 1} \tag{11}$$

The value of SCI varies between 0 and 1, 1 being the test sample can be represented as a linear combination of one family and 0 being the test sample is spread across all the families. It is common to have a threshold $\tau \in (0, 1)$ and reject outliers that are below τ.

For the Malimg Dataset, we vary τ for a fixed dimension ($D = 512$) as shown in Fig. 4a. At $\tau = 0.1$, the accuracy is 92.5% with no samples rejected. Accuracy of 100% is achieved when $\tau = 0.5$, at which 25% of the samples are rejected from the dataset. Similarly, for the Malheur dataset, we computed the accuracies and the percentage of samples dropped while varying τ. In Fig. 4b, we see that accuracy of 100% is reached when $\tau = 0.6$, but with only 5% of samples rejected.

4.4 Approximate l_1-norm

So far, we have used Basis Pursuit (BP) [7] for l_1-norm minimization and to recover the sparse coefficients. However, BP is computationally expensive and is not suitable for large scale data. Here, we compare the computation time and accuracy obtained using BP with an approximate l_1-norm minimization method, Orthogonal Matching Pursuit (OMP) [29]. OMP is a greedy algorithm that works by iteratively selecting a subset of columns from the training data matrix that are almost orthogonal. We repeat the experiments on both datasets using OMP and report the time taken to identify the families of all samples in the test set. The results are shown in Tab. 2. We see that for both datasets, the computation time decreased by a factor of 18 (Malimg) and 30 (Malheur) respectively, at the cost of slight decrease in classification accuracy. This makes OMP suitable for large scale malware classification.

4.5 Large Scale Analysis

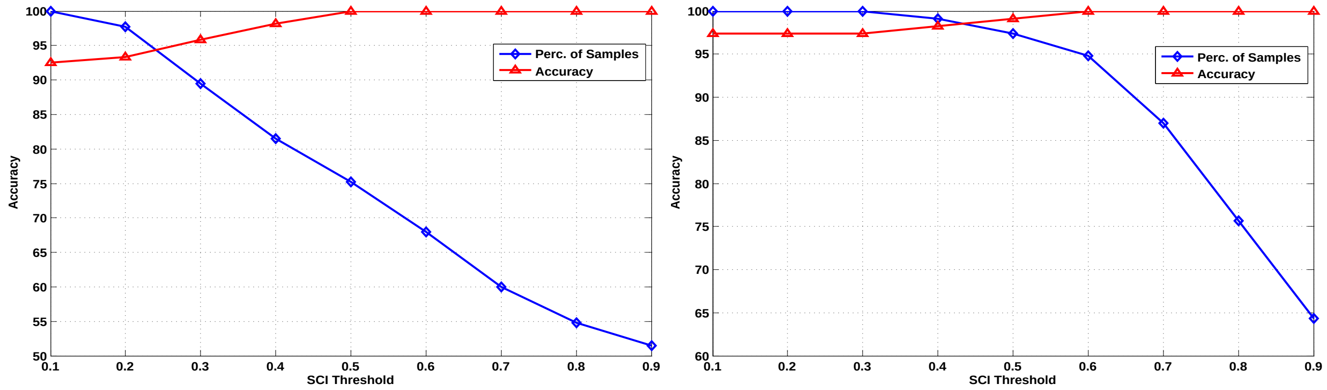
We evaluated our technique on two diverse large scale datasets. On both datasets, we randomly selected 20% of the data for testing and used Orthogonal Matching Pursuit to find the sparse coefficients. The results on both datasets show that our technique is applicable in large scale scenarios.

4.5.1 Results on Offensive Computing Dataset

We downloaded more than 1.4 Million malware from the Open Malware sharing platform [2] (formerly known as Offensive Computing). The samples were fed to different Antivirus software for labeling and the software that had minimum number of unknown labels was selected. This resulted in 2,124 malware families and we randomly selected 20 samples from each family to obtain a dataset of 42,480 samples (20 was the minimum number of samples present in some families). The size of the largest malware was 9.3 MB. We repeated the experiments using OMP and obtained an average classification accuracy of **66.34%**. The overall testing time was approximately 4 hours on a standard desktop machine. This time can further be reduced by using parallelization techniques. Out of 2,124 families, **927** families had an accuracy of **100%**. The average SCI value for these families was 0.97, with most values being 1. This shows that SCI can be used as a confidence measure during testing. At an SCI threshold of 0.6, 24.78% of the test samples were rejected and the classification accuracy was **77.08%**.

4.5.2 Results on Anubis Dataset

Next, we evaluated our technique on another large scale dataset that we obtained from the authors of Anubis [5]. The Anubis dataset had 36,784 samples divided into 209 clusters, with 176 samples per cluster. The clusters were labeled according to the behavioral pattern of a malware upon dynamic analysis [5]. This dataset is different from the Offensive Computing dataset in two aspects. First, the number of samples in a family/cluster is higher. Second, the labeling of clusters is based on dynamic analysis. This means there



(a) Malimg Dataset

(b) Malheur Dataset

Figure 4: Rejecting outliers based on Sparsity Coefficient Index (SCI). Higher the value of SCI, higher the classification accuracy. Both datasets achieve 100% accuracies at an SCI value of 0.6. For the Mallheur dataset, only 5% of samples are rejected to achieve this accuracy. However, for the Malimg dataset, nearly 32% of samples are rejected for the same.

Table 2: Basis Pursuit (BP) vs Orthogonal Matching Pursuit (OMP)

Dataset	BP Accuracy	OMP Accuracy	BP Computation Time (secs)	OMP Computation Time (secs)
Malimg Dataset	92.83	89.25	420	**24**
Malheur Dataset	98.55	97.39	180	**6**

is a possibility that two samples that have very different structure but similar behavior can be assigned the same cluster, and our technique will not work on such samples. For this dataset, the maximum size of the malware was 8.1 MB. On repeating the experiment, we obtained an average classification accuracy of **57.36%**. This is much lower than the accuracy obtained for the Offensive Computing dataset, which had more number of classes (by a factor of 10). This perhaps shows that our method is better applicable to malware datasets that have finer labels. The overall testing time was approximately 3 hours on a standard desktop. For this dataset, **27** clusters had an accuracy of **100%** and 50 clusters had an accuracy of more than 90%. On setting the SCI threshold to 0.6, 34.64% of the test samples were rejected and we obtained an accuracy of **77.12%**.

5. DISCUSSIONS AND CONCLUSION

Our approach works well mainly on malware variants that have similar structure. However, we observe that most variants are those that are structurally similar (for example, Regin variants in Fig. 1). This is also evident from our large scale experiments. Further, previous works such as [24] have shown that the performance of structurally similar features and behavior based features are almost the same.

In future, we will explore using Random Projections as malware signatures and distinguish them from benign samples. While our current approach estimates the family of a malware, we will also focus on identifying the exact source from which a malware variant evolves.

In this paper, we proposed a novel method to identify families of malware variants using a combination of Sparse Representation based Classification (SRC) and Random Projections (RP). Experiments on two standard malware datasets, as well as large scale data showed promising results. We believe that our approach, that is based on representing malware binaries as numerical signals, will open the scope of malware analysis to broader fields.

6. ACKNOWLEDGEMENTS

This work has been supported by grants ONR # N00014-11-10111 and ONR # N00014-14-1-0027.

7. REFERENCES

[1] Kaspersky lab is detecting 325,000 new malicious files every day. `http://usa.kaspersky.com/about-us/press-center/press-releases/kaspersky-lab-detecting-325000-new-malicious-files-every-day`.

[2] Offensive Computing Dataset. `http://offensivecomputing.net`.

[3] Regin: Top-tier espionage tool enables stealthy surveillance. `http://www.symantec.com/connect/blogs/regin-top-tier-espionage-tool-enables-stealthy-surveillance`.

[4] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan. N-gram-based detection of new malicious code. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, volume 2, pages 41–42. IEEE, 2004.

[5] U. Bayer, P. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, behavior-based malware clustering. In *Network and Distributed System Security Symposium (NDSS)*. Citeseer, 2009.

[6] V. Cevher, A. Sankaranarayanan, M. F. Duarte, D. Reddy, R. G. Baraniuk, and R. Chellappa. Compressive sensing for background subtraction. In *Computer Vision–ECCV 2008*, pages 155–168. Springer, 2008.

[7] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM journal on scientific computing*, 20(1):33–61, 1998.

[8] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu. Large-scale malware classification using random projections and neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3422–3426. IEEE, 2013.

[9] D. Donoho and J. Tanner. Counting faces of randomly projected polytopes when the projection radically lowers dimension. *Journal of the American Mathematical Society*, 22(1):1–53, 2009.

[10] J. Hegedus, Y. Miche, A. Ilin, and A. Lendasse. Methodology for behavioral-based malware analysis and detection using random projections and k-nearest neighbors classifiers. In *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, pages 1016–1023. IEEE, 2011.

[11] X. Hu, T. Chiueh, and K. Shin. Large-scale malware indexing using function-call graphs. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 611–620. ACM, 2009.

[12] G. Jacob, P. Comparetti, M. Neugschwandtner, C. Kruegel, and G. Vigna. A static, packer-agnostic filter to detect similar malware sample. In *Proceedings of the 9th Conference on Detection of Intrusions and Malware and Vulnerability Assessment*. Springer, 2012.

[13] J. Jang, D. Brumley, and S. Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 309–320. ACM, 2011.

[14] M. Karim, A. Walenstein, A. Lakhotia, and L. Parida. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, 1(1):13–23, 2005.

[15] D. Kirat, L. Nataraj, G. Vigna, and B. Manjunath. Sigmal: A static signal processing based malware triage. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, Dec 2013.

[16] J. Kolter and M. Maloof. Learning to detect and classify malicious executables in the wild. *The Journal of Machine Learning Research*, 7:2721–2744, 2006.

[17] J. Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, 3:91–97, 2006.

[18] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna. Polymorphic worm detection using structural information of executables. In *Recent Advances in Intrusion Detection*, pages 207–226. Springer, 2006.

[19] A. Lakhotia, A. Walenstein, C. Miles, and A. Singh. Vilo: a rapid learning nearest-neighbor classifier for malware triage. *Journal of Computer Virology and Hacking Techniques*, 9(3):109–123, 2013.

[20] P. Li, L. Liu, D. Gao, and M. K. Reiter. On challenges in evaluating malware clustering. In *Recent Advances in Intrusion Detection*, pages 238–255. Springer, 2010.

[21] X. Mei and H. Ling. Robust visual tracking and vehicle classification via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(11):2259–2272, 2011.

[22] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath. Malware images: visualization and automatic classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, VizSec '11, pages 4:1–4:7, New York, NY, USA, 2011. ACM.

[23] L. Nataraj, D. Kirat, B. Manjunath, and G. Vigna. Sarvam: Search and retrieval of malware. In *Proceedings of the Annual Computer Security Conference (ACSAC) Worshop on Next Generation Malware Attacks and Defense (NGMAD)*, 2013.

[24] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang. A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In *Proceedings of the 4th ACM workshop on Security and Artificial Intelligence*, AISec '11, pages 21–30, New York, NY, USA, 2011. ACM.

[25] A. Olivia and A. Torralba. Modeling the shape of a scene: a holistic representation of the spatial envelope. *Intl. Journal of Computer Vision*, 42(3):145–175, 2001.

[26] R. Perdisci and A. Lanzi. McBoost: Boosting scalability in malware collection and analysis using statistical classification of executables. *Computer Security Applications*, pages 301–310, Dec. 2008.

[27] J. K. Pillai, V. M. Patel, R. Chellappa, and N. K. Ratha. Secure and robust iris recognition using random projections and sparse representations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(9):1877–1893, 2011.

[28] K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19(4):639–668, 2011.

[29] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666, 2007.

[30] G. Wicherski. pehash: A novel approach to fast malware clustering. In *2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2009.

[31] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210–227, 2009.