SOPHOS

# Machine Learning With Feature Selection Using Principal Component Analysis for Malware Detection: A Case Study

Dr. Jason Zhang, *Sophos*

## ABSTRACT

Cybersecurity threats have been growing significantly in both volume and sophistication over the past decade. This poses great challenges to malware detection without considerable automation. In this paper, we have proposed a novel approach by extending our recently suggested artificial neural network (ANN)-based model with feature selection using the principal component analysis (PCA) technique for malware detection. The effectiveness of the approach has been successfully demonstrated with the application in PDF malware detection. A varying number of principal components is examined in the comparative study. Our evaluation shows that the model with PCA can significantly reduce feature redundancy and learning time with minimum impact on data information loss, as confirmed by both training and testing results based on around 105,000 real-world PDF documents. Of the evaluated models using PCA, the model with 32 principal feature components exhibits very similar training accuracy to the model using the 48 original features, resulting in around 33% dimensionality reduction and 22% less learning time. The testing results further confirm the effectiveness and show that the model can achieve 93.17% true positive rate (TPR) while maintaining the same low false positive rate (FPR) of 0.08% as the case when no feature selection is applied, which significantly outperforms all evaluated seven well known commercial antivirus (AV) scanners of which the best scanner only has a TPR of 84.53%.

**Keywords:** machine learning (ML), artificial neural network (ANN), multilayer perceptron (MLP), principal component analysis (PCA), cybersecurity, PDF malware, malicious documents, antivirus (AV)

## 1. INTRODUCTION

During the past decade, machine learning (ML), deep learning (DL) or generally termed artificial intelligence (AI) have gained wide popularity with applications across industries thanks to exponential improvement in computing hardware, availability of big datasets and improved algorithms. Recent work has witnessed breakthroughs from image classification, speech recognition to robot control and autonomous driving. In the meantime, cyber security attacks have been growing significantly in both volume and sophistication over the past decade as well. Typical examples include spamming campaigns, phishing emails and application vulnerability exploits, either via targeted or non-targeted attacks. It's no surprise that AI-based methods have been increasingly studied or applied in cyber security applications, varying from spam filtering,[1,2] intrusion detection,[3] malware detection[4] and classification.[5]

A crucial step in an ML workflow is feature extraction, which can be hand-crafted-based on human expertise, or automatically learned by training modern deep learning models such as convolutional neural networks (CNNs). It is natural to believe that more extracted features can provide better characterization of a learning task and more discriminating power. However, increasing the dimension of the feature vector could result in feature redundancy and noise. Redundant and irrelevant features can cause performance deterioration of an ML model with overfitting and generalization problems. Additionally, excessively increased number of features could significantly slow down a learning process. Therefore, it is of fundamental importance to only keep relevant features before feeding them into an ML model, which leads to requiring feature selection (or feature dimensionality reduction). Feature selection can be seen as the process of identifying and removing as much noisy and redundant information as possible from extracted features.

To demonstrate the effectiveness of our proposed ML approach with feature selection using PCA in malware detection, the suggested approach is evaluated with PDF malware detection as a case study.

Portable document format (PDF) is widely used for electronic documents exchange due to its flexibility and independence of platforms. PDF supports various types of data including texts, images, JavaScript, Flash, interactive forms and hyperlinks, etc. The popularity and flexibility of PDF also provide opportunities for hackers to carry out cyberattacks in various ways. A common PDF-based attack is through phishing, such as PDF-based order confirmation and parcel delivery notice, which typically uses social engineered texts to entice users to click phishing links embedded in PDF documents. Another common attack using PDF files: exploits targeting vulnerable PDF-reading applications. For example, an attack detected by Sophos targets four vulnerabilities of a popular PDF-reading application:[6] *Collab.collectEmailInfo* (CVE-2007-5659), *Util.printf* () (CVE-2008-2992), *Collab.getIcon* (CVE-2009-0927) and *Escript.api plugin media player* (CVE-2010-4091). Each of the exploits affects a particular version of the vulnerable application depending on the version installed on a victim's device.

There exist various approaches for the detection of PDF-based attacks. Typical traditional methods include signature match and sandbox-based analysis.[7–9] Given the popularity of ML in the past decade, the industry has also witnessed several ML applications in PDF malware detection, such as methods for detecting JavaScript in PDF files.[10,11] Other learning-based approaches for general PDF malware detection include a decision tree-based ensemble learning algorithm,[12] as well as methods using Naive Bayes, support vector machines (SVM) and Random Forest.[13–16] The pros and cons of the traditional methods and ML-based approaches above are discussed in.[17]

In this paper, we have proposed a novel approach by extending our recently proposed MLP neural network model $\text{MLP}_{\text{df}}$[17] with feature selection using PCA for malware detection. The performance is evaluated with PDF malware detection before and after applying feature selection. Feature selection with dimensionality reduction from $33\%$ up to $79\%$ is studied. Our evaluation based on real-world data shows that the model with PCA-based feature selection can significantly reduce feature redundancy and learning time with minimum impact on data information loss, as confirmed by both training and testing results.

The remainder of this paper is organized as follows: In the following section, the $\text{MLP}_{\text{df}}$ model from[17] is introduced. Section 3 discusses feature extraction, followed by feature selection using PCA. Then, Section 4 contains evaluation results illustrating the performance of the proposed approach with comparison to other methods. Finally, Section 5 contains our conclusions and suggested future work.

## 2. THE MLP$_{\mathrm{DF}}$ MODEL

The MLP$_{\mathrm{df}}$ neural network model we recently proposed in[17] has an input layer, an output layer, and two hidden layers between them. It is a densely connected network, which means each node in one layer fully connects to every node in the following layer. To carry out the supervised learning process, it is necessary to extract a digital representation $\boldsymbol{x}$ of a given object or event that needs to be fed into the MLP model. The learning task become finding a multidimensional function $\Psi(\cdot)$ which maps the input $\boldsymbol{x}$ to the target $\boldsymbol{y}$, as shown below

$$\boldsymbol{y} \cong \Psi(\boldsymbol{x}) \tag{1}$$

Where $\boldsymbol{x} \in \mathbb{R}^N$, a real-valued input feature vector $\boldsymbol{x} = [x_1, \cdots, x_N]^T$ in an $N$ dimensional feature space, with $(\cdot)^T$ denoting the transpose operation. Details on features and feature engineering will be discussed in Section 3. Similarly, $\boldsymbol{y} \in \mathbb{R}^M$ , a real-valued target classification vector $\boldsymbol{y} = [y_1, \cdots, y_M]^T$ in an $M$ dimensional classification space. In the application of PDF-based malware detection, $\boldsymbol{y}$ is a scalar ($M = 1$).

Fig. 1 depicts the architecture of the MLP$_{\mathrm{df}}$ model, where $\boldsymbol{x}$ and $\hat{y}$ denote input feature vector and trained binary output, respectively. $u_k^i$ denotes the neuron unit $k \in \mathbb{R}^{J,K}$ in the $i^{th}$ hidden layer, with $i = 1{,}2$ in this work. Each interconnection between the layers in the model is associated with a scalar weight $w_{j,k}$ which is adjusted during the training phase, where $j$ and $k$ are neuron unit indices between two consecutive layers.

The goal of the MLP learning process is to find a set of optimal weights $\{w_{j,k}\}$ over a training dataset in order to produce the outputs as close as possible to the target values. As Fig. 2 shows, the learning process with weights update is realized through the back-propagation (BP) algorithm which employs stochastic gradient decent (SGD) search through the space of possible weight values to minimize the error signal $e(y, \hat{y})$ between the trained output $\hat{y}$ and the target value $y$. In our training datasets, malicious PDF files are labelled with 1 and benign files are labelled with 0. The labels are then used as our target values$\{y\}$ for the training model.

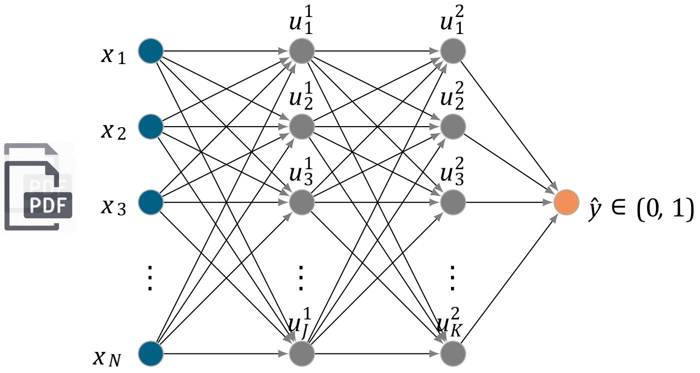

Figure 1. Architecture of the MLP$_{\mathrm{df}}$ Model[17]

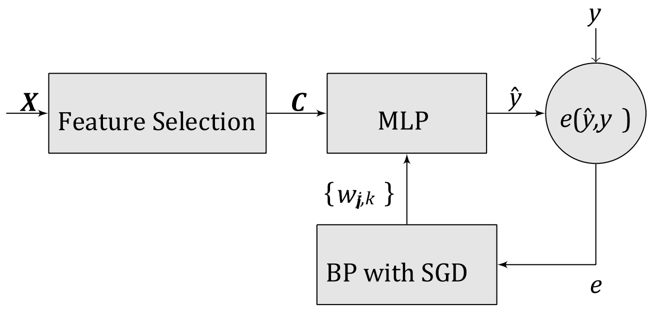

Figure 2. $\text{MLP}_{df}$ weights update via BP algorithm with SGD search.

More details on how the weights $\{w_{j,k}\}$ are updated during the learning process are discussed in.[17–19] In Fig. 2, there exists a process of feature selection between the original features $\boldsymbol{x}$ and the MLP model. This is to remove irrelevant features and only keep the most representative feature components $\boldsymbol{c}$. It is an important part of the so-called feature engineering process, as discussed in the following section.

## 3. FEATURE ENGINEERING

Feature engineering is a crucial step in machine learning. It is to make a task easier for an ML model to learn. One should not expect an ML model to be able to learn from completely arbitrary data. In many cases, data features rather than the raw data are used as input signals for an ML model. There is no exception when dealing with the detection of PDF attacks herein. Hand-crafted feature engineering exists using specific domain knowledge and automated feature engineering such as CNN-based models. The manual process of feature engineering is carried out in this work. It includes feature extraction from raw PDF files and feature selection using PCA techniques.

Table 1. Datasets and feature information

| **Training:** $90,000$ | | **Testing:** $15,047$ | | **Features:** $48$ |
|---|---|---|---|---|
| *Benign* | *Malicious* | *Benign* | *Malicious* | Structure, metadata Objects, content stats, etc. |
| 78,684 | 11,316 | 13,101 | 1,946 | |

### 3.1 Feature Extraction

The information for the training and testing datasets used in this work is shown in Table 1. An initial group of 48 features are extracted from the datasets. In-house tools and off-the-shelf PDF parsers are used to extract the features, which are comprised of information from PDF structure, metadata, object characteristics as well as content statistical properties, and more. Part of the 48 features are listed in Table 2. As it shows, features are defined using file size, JavaScript existence, page count, object count, stream filtering, entropy value of some content. The full list of features and the procedure of extraction are not discussed in this paper due to commercial reasons.

Table 2. Part of the extracted features

| **Feature name** | **Description** |
|---|---|
| *F SIZE* | *PDF file size* |
| *F JS* | *PDF with JavaScript or not* |
| *F PGC* | *Page count* |
| *F OBJC* | *Number of objects* |
| *F FILT* | *Stream filtering* |
| *F ENTRP1* | *Entropy of some content* |
| *F ENTRP2* | *Entropy of some content* |
| ... | ... |

As part of data pre-processing, all features need to be normalized before any further processes, such as feature selection and training. The primary aim of normalization is to avoid large gradient updates during the SGD search and learning process. Let

$$\boldsymbol{X} = [\boldsymbol{x}_1, \cdots \boldsymbol{x}_S]^T \qquad (2)$$

where $\boldsymbol{X} \in \mathbb{R}^{S \times N}$ is a feature matrix, $\boldsymbol{x}$ a feature vector defined in (1) and $S$ the size of a malware or benign dataset. Typically, normalization is applied for each feature independently. More specifically, this is carried out along each column vector of $\boldsymbol{X}$. The normalized feature vector then has a standard normal (or Gaussian) distribution with $\mu = 0$ and $\sigma = 1$ where $\mu$ and $\sigma$ are the mean and standard deviation of the scaled feature vector, respectively.

Though we believe that the features extracted herein should possess strong discriminative power to differentiate malicious files from benign documents, it is unavoidable to have redundancy among the features. This is why a feature selection process is often applied prior to a learning process.

### 3.2 Feature Selection Using PCA

Irrelevant and redundant features can lead to an ML classifier to converge slowly and perform less well or completely fail. In this paper, the PCA technique (also known as the eigenvector regression filter or the Karhunen-Loeve transform[20]) is used for dimensionality reduction, which involves zeroing out one or more of the weakest principal components, resulting in a lower-dimensional projection of the raw feature data that preserves the maximal data variance. The dimensionality reduction process is achieved through an orthogonal, linear projection operation. Without loss of generality, the PCA operation can be defined as

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{C} \qquad (3)$$

with $\boldsymbol{Y} \in \mathbb{R}^{S \times P}$is the projected data matrix that contains *P* principal components of $\boldsymbol{X}$ with$P \leq N$. So the key is to find the projection matrix $\boldsymbol{C} \in \mathbb{R}^{N \times P}$, which is equivalent to find the eigenvectors of the covariance matrix of $\boldsymbol{X}$, or alternatively solve a singular value decomposition (SVD) problem for $\boldsymbol{X}$ [19]

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{4}$$

where $\boldsymbol{U} \in \mathbb{R}^{S\times S}$ and $\boldsymbol{V} \in \mathbb{R}^{N\times N}$ are the orthogonal matrices for the column and row spaces of $\boldsymbol{X}$, and $\boldsymbol{\Sigma}$ is a diagonal matrix containing the singular values, $\lambda_n$, for $n = 0, \cdots, N-1$, non-increasingly lying along the diagonal. It can be shown[19] that the projection matrix **C** can be obtained from the first *P* columns of $\boldsymbol{V}$ with

$$\mathbf{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N] \tag{5}$$

and

$$\mathbf{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_P] \tag{6}$$

where $\boldsymbol{v}_n \in \mathbb{R}^N$ is the $n^{th}$ right singular vector of $\boldsymbol{X}$, and $\boldsymbol{c}_n = \boldsymbol{v}_n$.

In fact, the singular values contained in $\boldsymbol{\Sigma}$ in (4) are the standard deviations of $\boldsymbol{X}$ along the principal directions in the space spanned by the columns of $\mathbf{C}$.[19] Therefore, $\lambda_n^2$ becomes the variance of $\boldsymbol{X}$ projection along the $n^{th}$ principal component direction. It is believed that variance can be explained as a measurement of how much information a component contributes to the data representation. One way to examine this is to look at the cumulative explained variance ratio of the principal components, given as

$$R_{cev} = \frac{\sum_{n=1}^{P} \lambda_n^2}{\sum_{n=1}^{N} \lambda_n^2} \tag{7}$$

and illustrated in Fig. 3. It indicates that keeping only a few principal components could retain over $90\%$ of the full variance or information of $\boldsymbol{X}$. As a comparative study, a varying number of principal components has been used and examined in the following evaluation section.

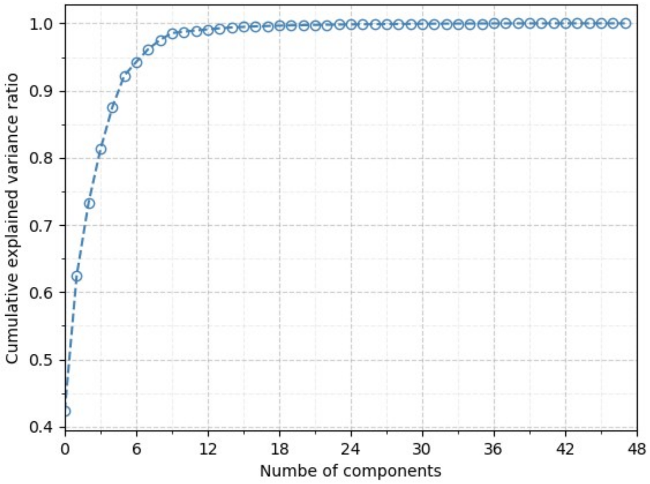

Figure 3. Cumulative explained variance ratio over components.

## 4. EVALUATION RESULTS

The evaluation is based on the datasets described in Table 1, which comprises $105{,}000$ real-world benign and malicious PDF documents collected from Sophos database. The malicious PDF documents were mostly collected over the few months up to March 2018 while the benign dataset has a longer timespan.

It is of fundamental importance to tackle the overfitting problem in any ML process. There is no exception in our application. A trained ML algorithm must perform well with new data which was never seen during training process. Typical techniques to mitigate overfitting include *Batch normalization* and *Dropout*.[18,19] To make the evaluated models learn better during training and generalize well on new data, batch normalization with a batch size of 64 data points is applied to the input of each layer after the input layer. This is to re-scale the input batch to have zero mean and unit variance, similar to the feature normalization discussed in Section 3.1. Similarly, a dropout rate of 0.15 is used during training process, which leads $15\%$ of each hidden layer outputs to be zeroed out before feeding into next layer. In addition, around $20\%$ of the training dataset is used as the validation dataset to help detect overfitting and perform model selection during the learning process.

In our previous work,[17] the input layer of the $\text{MLP}_{df}$ model has 48 nodes corresponding to the number of original features used, and the output layer has a single Sigmoid (binary) probability output with values in the range of $(0,1)$. There are two hidden layers with 72 neurons each. In this evaluation, the number of input nodes for the models with PCA-based feature selection will be dependent on the selected number of principal feature components, the rest of the model settings remains the same as the $\text{MLP}_{\text{df}}$ model. A varying number of principal components is used in the comparative study (resulting in dimensionality reduction of $79\%$ , $41\%$ and $33\%$ ), and we term the corresponding models as $\text{MLP}_{\text{df}}$+PCA10, $\text{MLP}_{\text{df}}$+PCA28 and $\text{MLP}_{\text{df}}$+PCA32, respectively. These models are trained with $5{,}000$ *epoch*s each. An *epoch* refers to a complete training cycle and many *epoch*s are needed in order to accomplish an ANN training task. The training results are compared in Fig. 4. As the figure shows, all models quickly reach over $97\%$ accuracy after a small amount of training epochs, then the accuracies continue to improve with relatively slow convergence rates. It indicates that using the first 10 principal feature components could achieve around 98% training accuracy after $3{,}000$ Epochs, this is consistent with the observation of the cumulative explained variance ratio shown in Fig. 3. It is no surprise that the model with 28 principal feature components performs even better. As the figure shows, the best model using PCA is $\text{MLP}_{\text{df}}$+PCA32 which exhibits excellent accuracy similar to the original $\text{MLP}_{\text{df}}$ model while with around 33% dimensionality reduction and $22\%$ less learning time.

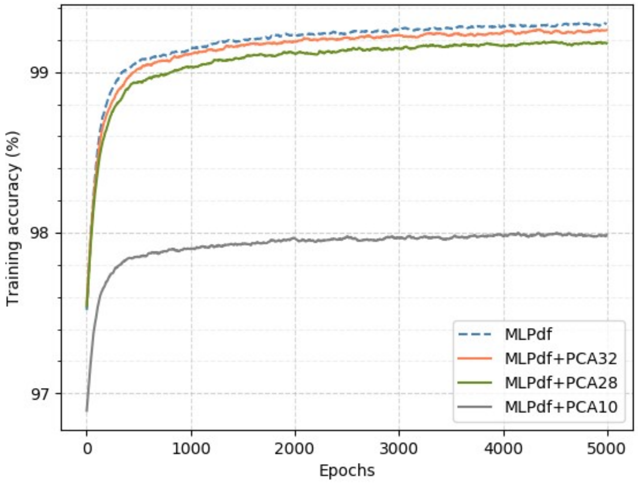

Figure 4. Training results comparison with/out feature selection.

To further examine the performance consistence and generalization of the model using PCA-based feature selection, the model MLP$_{df}$+PCA32 is tested with the testing dataset shown in Table 1, and compared with the original MLP$_{df}$ model as well as other seven major commercial AV scanners. The comparison results are shown in Fig. 5 where AV1 to AV7 denote the corresponding commercial AV scanners. The best result is from MLP$_{df}$ which achieves an excellent 95.12% TPR while maintaining a very low FPR of 0.08%, as discussed in.[17] This is closely followed by the proposed MLP$_{df}$+PCA32 which manages to reach 93.17% TPR with the same low FPR of 0.08% maintained. Both models significantly outperform all commercial scanners. The best commercial scanner (denoted as AV4) only has a TPR of 84.53%.

It is worth pointing out that the evaluated seven commercial scanners perform well with zero or very low FPR with the benign testing dataset as well. One possible reason could be that our benign files are mostly collected from Sophos clean PDF documents database which has a relatively longer timespan, and the majority of them might have been already known to the commercial scanners as well.

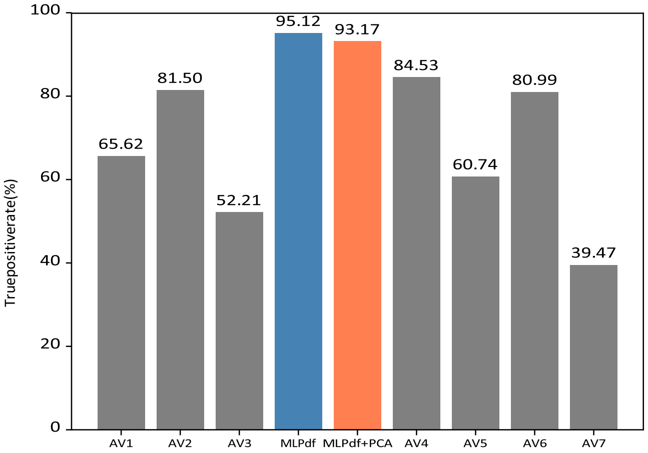

Figure 5. Testing results between MLP$_{df}$ and major AV scanners.

## 5. CONCLUSION

In this paper, we have proposed a novel approach by extending our recently proposed ANN model $\text{MLP}_{\text{df}}$ in[17] with feature selection using PCA technique for malware detection. As a case study, the effectiveness of the approach has been successfully demonstrated with the application in PDF malware detection. Our evaluation shows that the model with PCA can significantly reduce feature redundancy with minimum impact on data information loss, as confirmed by both training and testing results based on around $105{,}000$ real-world PDF documents. More specifically, a comparative study has been carried out for the model with a varying number of selected principal feature components. Of the evaluated models using PCA, the model with 32 principal feature components, termed $\text{MLP}_{\text{df}}$+PCA32, exhibits very similar training accuracy to the $\text{MLP}_{\text{df}}$ model while with around $33\%$ dimensionality reduction and $22\%$ less learning time. The testing results further confirm the effectiveness and show that the proposed $\text{MLP}_{\text{df}}$+PCA32 model is able to achieve $93.17\%$ TPR while maintaining the same low FPR of $0.08\%$ as the case with $\text{MLP}_{\text{df}}$. On the other hand, the best commercial scanner (denoted as AV4) only manages to have a TPR of $84.53\%$. As previously pointed out, the commercial scanners perform well on the benign testing files as well with zero or very low FPR. It will be interesting to compare how the ANN-based models and other commercial scanners perform with larger datasets in our future work, particularly adding more recent PDF documents to the benign corpus. Given the fact that PCA is restricted to a linear transformation, it is worth exploring non-linear feature selection techniques such as autoencoders.

## REFERENCES

[1] C.-H. Wu, "Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks," *Expert Systems with Applications* **36(3)**, pp. 4321–4330, 2009.

[2] S. Kumar, X. Gao, I. Welch, and M. Mansoori, "A Machine Learning Based Web Spam Filtering Approach," in *Proceedings of IEEE 30th International Conference on Advanced Information Networking and Applications*, (Crans-Montana, Switzerland), 2016.

[3] V. Engen, *MACHINE LEARNING FOR NETWORK BASED INTRUSION DETECTION*. PhD thesis, Bournemouth University, 2010.

[4] Sophos, "Sophos Unmatched next-gen endpoint protection: Intercept-X." https://www.sophos.com/en-us/products/intercept-x.aspx. Accessed: 2018-03.

[5] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *Proceedings of 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP*, (Brisbane, Queensland, Australia), 2015.

[6] J. Zhang, "Make "Invisible" Visible - Case Studies in PDF Malware," in *Proceedings of Hacktivity 2015*, (Budapest, Hungary), 2015.

[7] Z. Tzermias, G. Sykiotakis, M. Polychronakis, and E. P. Markatos, "Combining Static and Dynamic Analysis for the Detection of Malicious Documents," in *Proceedings of the fourth Workshop on European Workshop on System Security*, (Salzburg, Austria), 2011.

[8] P. Ratanaworabhan, B. Livshits, and B. Zorn, "NOZZLE: A Defense Against Heapspraying Code Injection Attacks," in *Proceedings of the 18th conference on USENIX security symposium*, (Berkeley, CA USA), 2009.

[9] C. Willems, T. Holz, and F. Freiling, "Toward Automated Dynamic Malware Analysis Using CWSandbox," *IEEE Security & Privacy* **5(2)**, 2007.

[10] Wepawet. http://wepawet.iseclab.org/. Accessed: 2018-03.

[11] P. Laskov and N. Srndic, “Static Detection of Malicious JavaScript-Bearing PDF Documents,” in *Proceedings of the 27th Annual Computer Security Applications Conference*, (Orlando, Florida USA), 2011.

[12] J. S. Cross and M. A. Munson, “Deep pdf parsing to extract features for detecting embedded malware,” Tech. Rep. SAND2011-7982, Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, CA 94550, Sept. 2011.

[13] D. Maiorca, G. Giacinto, and I. Corona, “Machine Learning and Data Mining in Pattern Recognition,” in *volume 7376 of Lecture Notes in Computer Science*, (Springer Berlin / Heidelberg), 2012.

[14] C. Smutz and A. Stavrou, “Malicious PDF Detection using Metadata and Structural Features,” in *Proceedings of the 28th Annual Computer Security Applications Conference*, (Orlando, Florida USA), 2012.

[15] N. Srndic and P. Laskov, “Detection of Malicious PDF Files Based on Hierarchical Document Structure,” in *Proceedings of the 20th Annual Network & Distributed System Security Symposium*, (San Diego, CA USA), 2013.

[16] B. Cuan, A. Damien, C. Delaplace, and M. Valois, “Malware Detection in PDF Files Using Machine Learning,” Tech. Rep. Rapport LAAS No. 18030, REDOCS, Feb. 2018.

[17] J. Zhang, “MLP$_{df}$: An Effective Machine Learning Based Approach for PDF Malware Detection,” in *Black Hat USA 2018*, (Las Vegas, NV, USA), 2018.

[18] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, The MIT Press, 2016.

[20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, 2nd Edition*, Addison Wesley, 2002.

SOPHOS