Poster: Ensemble Learning-Based Detection of Office Malware

1st Michael Blair
Oracle Labs
Brisbane, Australia
michael.blair@live.com.au

2nd François Gauthier
Oracle Labs
Brisbane, Australia
francois.gauthier@oracle.com

3rd Scott Gaetjen
Oracle
Frisco, Texas
scott.gaetjen@oracle.com

***Abstract*—The initial step of many cyberattacks is to infect a target through a suitable infection vector. Among the plethora of vectors available to attackers, email attachments remain a common and cost-effective way to perform initial infection. Of all the kinds of attachments that can be used to initially infect a host, Office documents became significantly more common in 2018. Indeed, according to the 2019 Internet Security Threat Report [1], 48% of malicious email attachments were Office files in 2018, up from 5% in 2017. In this work, we introduce a novel ensemble-learning approach that detects Office malware based on three sources of information: text content, metadata, and Visual Basic for Applications (VBA) macro code. Contrary to previous studies that focus on detecting malicious VBA macros only, our ensemble learning approach can also detect macro-less Office malware, such as social engineering or phishing documents, which account for a large proportion of malicious email attachments in our dataset. Learning from human-interpretable features further improves the overall explainability of our classifier. Evaluation on a dataset of 32 450 open-source benign and 11 551 internally sourced malicious documents shows that our ensemble classifier achieves a precision of 97% and a recall of 98%.**

***Index Terms*—Malware, Office documents, Ensemble learning, Explainability**

I. Introduction

Among the various potential infection vectors in internet-facing systems, human actors remain a vector of choice. Indeed, Symantec's 2019 Internet Security Threat Report [1] reveals that one in 412 emails was malicious in 2018, and that over 90% of them used attachments, which require human intervention, as payloads. Furthermore, the report also reveals that 48% of those attachments were Office files, up from 5% in 2017. To address this increasing threat, we present an ensemble architecture for detecting malicious Office documents. This architecture combines three distinct sources of information in a novel way and uses document metadata features, which have been overlooked in previous work. By extracting features from document text and metadata in addition to VBA code, this new architecture broadens the scope of detectable malware well beyond that of previous VBA macro-based detectors. Using an ensemble of classifiers to make predictions based on text, metadata and VBA features enhances interpretability by providing insight into which information source contains 'suspicious' content. Furthermore, using "traditional" classifiers (e.g. decision tree, random forests, support vector machine, naïve Bayes, and k-nearest neighbours) allows for effortless investigation into the features that lead to a malicious classification, in contrast to popular yet notoriously difficult to interpret Nerual Network models, while retaining high performance. This new architecture achieves convincing results by precisely and reliably detecting a range of real-world malicious Office documents.

II. Methodology

A. Dataset & Features

32 450 benign samples consisting of de-duplicated Office documents from the Enron email attachment dataset and 11 551 internal email attachments flagged as malicious by Oracle's security team were sourced. As outlined in Figure 2, upon analysing an Office sample for the first time, our architecture first extracts text and raw metadata information using the Oracle Outside In technology [2], followed by VBA code extraction using Apache POI [3]. Raw **text** is represented as vectors of Term Frequency Inverse Document Frequency (TFIDF) scores. Semi-structured **metadata** properties undergo a variety of transformations, such as one-hot encoding of categorical properties, scaling for continuous properties, and abstracting strings and arrays to their length. Feature engineering was automated using a custom 'transformer', performing engineering steps after a small range of statistical measures on each feature. **VBA** code is tokenized, using an custom-built lexer based on the VBA language specification [4]. Token streams are converted to token frequency vectors by using the bag-of-words technique. Most relevant categorical and continuous features are selected with Chi-square analysis, and Analysis of Variance (ANOVA) respectively.

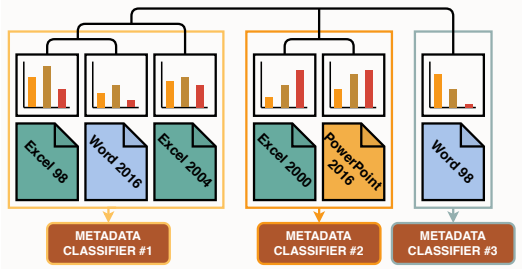

Fig. 1: HCA on Feature Distribution to Create More Specific Classifiers

B. Clustering on Feature Distribution

Experiments showed that training a single metadata classifier, irrespective of file types, yields poor predictions due to a large imbalance in the metadata properties of each file type. To overcome this limitation, we group file types together using Hierarchical Cluster Analysis based on their metadata feature distribution, and train individual classifiers on each cluster which are selected at prediction time based on a sample's file format. Following clustering, the dataset was curated by: 1. balancing clusters to obtain a 50% class split, and 2. only keeping clusters with a significant number of documents.

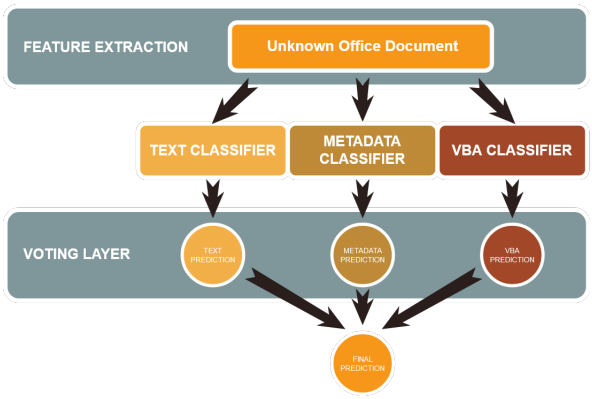

Fig. 2: Ensemble learning-based Office malware detection

C. Architecture

Once features are extracted, engineered, and selected, we select the best performing classifiers from amongst a collection indicated in Table I, which are then tuned by employing a grid search strategy. To aggregate the individual predictions, we use a voting layer, as shown in Figure 2. We trialled multiple voting methods and found that biased voting results in the best ensemble performance. In this scheme, the ensemble mimics the output of the classifier with the highest accuracy, unless outvoted by both of the other classifiers.

TABLE I: Average F1 scores with default hyperparameters after 10-fold cross-validation

Classifier	Text F1	Metadata F1	VBA F1
K-Nearest Neighbours	0.91	0.89	0.88
Support Vector Machine	0.38	0.77	0.92
Decision Tree	0.88	0.90	**0.98**
Naïve Bayes	0.9	0.87	0.95
Random Forest	**0.92**	**0.92**	0.95

D. Explainability

Predictions from individual classifiers in the ensemble already hint at the 'suspicious' components of a document. To gain additional insight, an analyst can further investigate the feature weights of each classifier to find the most influential features. Table II shows the most influential features in our dataset. For example, the term 'powershell' in document text, or the `SHELL` or `AUTO OPEN` tokens in the VBA code are strong indicators of maliciousness that are rarely, if ever, present in benign documents, leading to such a high precision in the VBA classifications. Interestingly, the frequency of the `PLUS` token (+) in VBA code is indicative of string concatenation obfuscation, a technique that isn't inherently malicious, but in practice was not employed in benign samples. Metadata properties such as the number of words, paragraphs or revisions (saved changes made to the document) along with the term usage frequency of countries, currencies (e.g. "$5.00") or people's names are all discriminating features.

TABLE II: Most influential features per classifier

Classifier	Features
Text	powershell, time, company, information, trading, meeting
VBA	`SHELL_API`, `VB_GLOBALNAMESPACE`, `FOR`, `AUTOOPEN_AUTO`, `CREATEOBJECT_API`, `PLUS`
Metadata	count_lines, word_count, count_paras, country_count, currency_count, last_saved_by_length, person_name_length

III. Results & Evaluation

TABLE III: Predictive power of task-specific classifiers

Classifier	Train size	Test size	Precision	Recall	Accuracy
Text	6962	816	0.9772	0.9817	0.9779
VBA	2841	287	1.0	0.9957	0.9965
Metadata	13450	2000	0.9694	0.9790	0.9740
Ensemble	19906	2000	0.9694	0.9800	0.9745

Table III compares classifier predictive power after tuning, and also shows how out of the 19 906 train and 2 000 test documents, many lack either text, VBA code, or metadata, as reflected in the train and test set sizes. This suggests that previous works relying on VBA features only [5], [6] would fail to detect a high number of malicious documents in practice. By using three sources of information, our ensemble classifier extends the range of detectable malware and achieves a **precision of 97%** and **recall of 98%**.

References

[1] Symantec, "Internet security threat report," https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf, 2019, accessed: 30-03-2020.

[2] Oracle, "Outside In Clean Content," https://www.oracle.com/technetwork/middleware/content-management/oit-all-085236.html, accessed: 30-03-2020.

[3] A. S. Foundation, "Apache POI - the Java API for Microsoft Documents," https://poi.apache.org/, accessed: 30-03-2020.

[4] Microsoft, "VBA Language Specification," https://docs.microsoft.com/en-us/openspecs/microsoft_general_purpose_programming_languages/ms-vbal/d5418146-0bd2-45eb-9c7a-fd9502722c74, accessed: 30-03-2020.

[5] E. Aboud and D. O'Brien, "Detection of malicious VBA macros using Machine Learning methods," in *Proceedings for the 26th AIAI Irish Conference on Artificial Intelligence and Cognitive Science*. CEUR-WS.org, 2018, pp. 374–385.

[6] S. Kim, S. Hong, J. Oh, and H. Lee, "Obfuscated VBA macro detection using machine learning," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 490–501.