ALDOCX: Detection of Unknown Malicious Microsoft Office Documents Using Designated Active Learning Methods Based on New Structural Feature Extraction Methodology

Nir Nissim, Aviad Cohen, and Yuval Elovici

***Abstract*— Attackers increasingly take advantage of innocent users who tend to casually open email messages assumed to be benign, carrying malicious documents. Recent targeted attacks aimed at organizations utilize the new Microsoft Word documents (∗.docx). Anti-virus software fails to detect new unknown malicious files, including malicious docx files. In this paper, we present ALDOCX, a framework aimed at accurate detection of new unknown malicious docx files that also efficiently enhances the framework's detection capabilities over time. Detection relies upon our new structural feature extraction methodology (SFEM), which is performed statically using meta-features extracted from docx files. Using machine-learning algorithms with SFEM, we created a detection model that successfully detects new unknown malicious docx files. In addition, because it is crucial to maintain the detection model's updatability and incorporate new malicious files created daily, ALDOCX integrates our active-learning (AL) methods, which are designed to efficiently assist anti-virus vendors by better focusing their experts' analytical efforts and enhance detection capability. ALDOCX identifies and acquires new docx files that are most likely malicious, as well as informative benign files. These files are used for enhancing the knowledge stores of both the detection model and the anti-virus software. The evaluation results show that by using ALDOCX and SFEM, we achieved a high detection rate of malicious docx files (94.44% TPR) compared with the anti-virus software (85.9% TPR)—with very low FPR rates (0.19%). ALDOCX's AL methods used only 14% of the labeled docx files, which led to a reduction of 95.5% in security experts' labeling efforts compared with the passive learning and the support vector machine (SVM)-Margin (existing active-learning method). Our AL methods also showed a significant improvement of 91% in number of unknown docx malware acquired, compared with the passive learning and the SVM-Margin, thus providing an improved updating solution for the detection model, as well as the anti-virus software widely used within organizations.**

***Index Terms*— Active learning, machine learning, structural, documents, microsoft office files, docx, malware, malicious.**

Manuscript received March 16, 2016; revised August 6, 2016 and September 30, 2016; accepted November 11, 2016. Date of publication December 1, 2016; date of current version January 18, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mauro Conti.

The authors are with the Malware Laboratory, Cyber Security Research Center, Department of Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel (e-mail: nirni@post.bgu.ac.il).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2016.2631905

I. Introduction

Since 2009, cyber-attacks against organizations have increased, and 91% of all organizations were hit by cyber-attacks in 2013.[1] The vast majority of organizations rely heavily on email for internal and external communication, and email has become a very attractive platform by which to initiate cyber-attacks against organizations [17]. Attackers usually use social engineering in order to encourage recipients to press a link or open a malicious web page or attachment. According to *TrendMicro*,[2] attacks, particularly those against government agencies and large corporations, are almost entirely launched through spear-phishing emails. An incident in 2014 aimed at the Israeli Ministry of Defense (IMOD) provides an example of a new type of targeted cyber-attack involving a document attached to an email. According to media reports,[3] the attackers posed as IMOD representatives and sent email messages with a malicious document attachment which installed a Trojan horse once it was opened—enabling the attacker to control the compromised computer. This type of attack which relies on non-executable documents has grown in popularity, partly because executable files attached to emails are filtered out by most email servers due to security reasons. In addition, most users today consider non-executable files safer than executable files, and therefore users are less suspicious of such files. However, non-executable files are as dangerous as executable files, because their readers may contain vulnerabilities that can be exploited for malicious purposes by attackers. Cybercriminals launch attacks through Microsoft Office files,[4] taking advantage of the fact that Office documents are widely used among most organizations. Cybercriminals exploit the fact that most employees within organizations do not take precautions when receiving and opening these files.

In an attempt to prevent or mitigate these types of attacks, Microsoft Office provides several mechanisms to try to reduce the possibility of attack, including mechanisms such as macro

[1]http://www.humanipo.com/news/37983/91-of-organisations-hit-by-cyber attacks-in-2013/

[2]http://www.infosecurity-magazine.com/view/29562/91-of-apt-attacks-start-with-a-spearphishing-email/

[3]http://uk.reuters.com/article/2014/01/26/us-israel-cybersecurity-idUKBREA0P0ON20140126

[4]http://securelist.com/blog/research/65414/obfuscated-malicious-office-documents-adopted-by-cybercriminals-around-the-world/

security level, trusted locations, and digital signatures. However, Dechaux *et al.* [2] showed that these mechanisms can be easily bypassed. Dechaux et.al explored different weaknesses within Microsoft Office and OpenOffice which allow potential attackers to bypass default security settings that prevent macros from being executed by default when a file is opened. In addition to macro execution bypass, they revealed how to build an attack that requires neither sophisticated techniques nor special knowledge while remaining totally undetected by anti-virus software. As a proof of concept, they implemented two attacks for both Microsoft Office and OpenOffice. The first attack changes the security level to the lowest level. The second attack adds a new trusted location path. In these attacks a malicious macro code can be executed, despite the security mechanisms in place.

Many studies have been focused on the detection of malicious PDF documents, as was surveyed by Nissim *et al.* [8]. The closest work to our study in terms of methodology is presented in [14]. Even if the authors use an hierarchy file structures approach, similar to our approach, this study focuses on the PDF file and is not general enough to be also applied to others file structures such as MS office documents. PDF files differ significantly from docx files, in two important ways that highlight the novelty and contributions made by our framework. First, PDF files consist of a set of related objects, while docx files are archival files consisting of folders and XML files. Given this, applying a structural based approach for malicious PDF detection [14] on malicious docx files would be an unsuccessful detection approach. Second, PDF and docx files are associated with different attack techniques, and even when they do share an attack technique (e.g., embedded-files), the attack is launched very differently, such that the same attack affects the file structure differently in each case. Therefore, the detection of malicious docx files requires different and customized structural features based on the special file structure of these files.

Keeping pace with the continuous creation of benign Microsoft Office files by organizations and their use with emails, new malicious Microsoft Office files and variants of existing malicious files, designed to evade the detection of signature-based anti-virus, are also being created on an ongoing basis. Thus, in order to effectively analyze tens of thousands of new, potentially malicious Microsoft Office files on a daily basis, anti-virus vendors have integrated a component of a detection model based on machine learning and rule-based algorithms [13] into the core of their signature repository update activities. However, these solutions are often ineffective, because their knowledge base is not adequately updated. This occurs because an enormous number of new, potentially malicious Microsoft Office files are created each day, and only a limited number of security researchers are tasked with manually inspecting and labeling them. Thus, there is a need to prioritize which files should be acquired, analyzed, and labeled by a human expert. As Microsoft Office Word files are the most popular Microsoft Office files used by organizations, our work focuses on this type of file.

In this study we present ALDOCX – a framework and new structural feature extraction methodology (SFEM) that is aimed at the accurate detection of malicious Microsoft Word XML-based (∗.docx) files.

This paper is based on our preliminary results which were presented in a recently published study [16]; the current work is a comprehensive extension of this research accompanied by additional experimental evaluations, and provides further in depth analysis and additional results and insights.

ALDOCX is an active learning based framework for frequently updating the detection model with docx files. ALDOCX focuses on improving the detection mode by labeling the docx files (potentially malicious or informative benign files) that are most likely to improve the detection model's performance and, in so doing, enrich the signature repository with as many new malicious docx files as possible, thus enhancing the detection process. Specifically, the ALDOCX framework favors files that contain new content. In this paper we focus on PCs and docx files, the platform and documents most used by organizations. Microsoft Word legacy binary files (∗.doc) are beyond the scope of this paper as they have a file structure that substantially differs from the new ∗.docx files (docx) and requires a feature extraction methodology that is specifically tailored for them. Our contributions in this paper are:

- Presenting SFEM, new methodology of features extraction based on structural paths within a docx file capable of providing accurate detection of malicious docx files.
- Presenting the use of machine learning algorithms for the detection of malicious Microsoft Word documents based on the new structural feature extraction methodology (SFEM).
- Developing ALDOCX, a framework based on active learning methods for enhancing and maintaining detection capabilities and updatability.

II. Background

Microsoft Office 97 and later versions used the legacy binary file as their default file format, which is referred to as "Microsoft Office 97-2003." These files can be read only using Microsoft Office. The extensions for the well-known Microsoft Word, Excel, and PowerPoint files are ∗.doc, ∗.xls, and ∗.ppt respectively. With the release of Microsoft Office 2007, Microsoft introduced an entirely new file format based on XML called "Open XML."[5] The new file format applies to Microsoft Word, Excel, and PowerPoint and comes with the addition of the "x" suffix to the recognized file extension: ∗.docx, ∗.xlsx, and ∗.pptx. The files are automatically compressed (up to 75% in some cases) using *Zip* technology, and when the file is opened, it is automatically unzipped.

In 2013, Schreck *et al.* [1] from Siemens Cert presented a new approach called *BISSAM (Binary Instrumentation System for Secure Analysis of Malicious Documents)* with three purposes: distinguishing malicious documents from benign documents, extracting embedded malicious shellcode, and detecting and identifying the vulnerability exploited by a malicious document. The system inspects the Office file, extracts the

[5]http://office.microsoft.com/en-001/help/introduction-to-new-file-name-extensions-HA010006935.aspx

embedded malicious shellcode from it, and then automatically determines what type of vulnerability the malicious code targets. The first type of vulnerability is that for which an update patch already exists. The second type is a new security flaw which requires deeper analysis, and in this context the system also provides information about the specific vulnerability exploited and which patch is required. The system focuses on Microsoft Office 2003 and 2007 running on Microsoft Windows XP, but it can be also used with other applications. A number of other tools aimed at detecting malicious Office files exist. *OfficeMalScanner*[6] is a free forensic tool to scan for malicious traces, such as shellcode heuristics, PE-files or embedded OLE streams in legacy binary Microsoft Office files. Another tool based on vulnerability signatures is *SourceFire's*[7] product called *OfficeCat*[8] which is a file checker based on vulnerability signatures, aimed at the detection of various vulnerabilities in Microsoft Office legacy binary files. Microsoft *OffVis,*[9] a free tool that provides visualization for displaying the raw content and structure of Microsoft Office legacy binary documents and identifies some common exploits, is also based on vulnerability signatures. *pyOLEScanner.py*[10] is a Python based script that can examine and decode some aspects of malicious legacy binary Microsoft Office files.

In addition to the tools mentioned above, there are several tools that perform dynamic analysis on files. These tools are capable of analyzing both executable and non-executable files, such as ∗.exe, ∗.pdf, ∗.docx, etc. The analysis process is done by executing the suspicious file in an isolated environment and examining the effect of the behavior and actions on the system during runtime. The *CheckPoint* company's*Threat Emulation*[11] product, executes the suspicious file in multiple operating systems and with different versions of viewing programs (such as Microsoft Office or Adobe Reader), and if the tool detects that an unusual network connection has been established or changes have been made to the File System, Registry, or processes, the file will be labeled as malicious. However, dynamic analysis has several disadvantages, including its high resource costs, computational complexity, and the time it requires; in addition, it can also be detected by the executed malicious code and consequently cease its malicious operations. On the other hand, static analysis methods have several advantages over dynamic analysis. First, they are virtually undetectable—the docx file cannot detect that it is being analyzed, since it is not executed. While it is possible to create static analysis "traps" to deter static analysis, these traps can themselves be leveraged and used to detect malware using static analysis. In addition, since static analysis is relatively efficient and fast, it can be performed in an acceptable timeframe and consequently will not cause bottlenecks. Static analysis is also easy to implement, monitor, and measure. Moreover, it scrutinizes the file's "genes" and not its current behavior which can be changed or delayed to an unexpected time or specific conditions in order to evade detection by the dynamic analysis. Static analysis can also be used for a scalable pre-check of malwares.

All the above mentioned tools can be categorized into two groups: the first is aimed at detecting only Microsoft Office legacy binary files. The second group is also capable of analyzing the new XML-based format files, however it does this through dynamic analysis with all its disadvantages. Both of these groups of tools share the same, significant disadvantage in that they use deterministic analysis and/or rule based heuristics in order to detect maliciousness. Nevertheless, none of them apply machine learning algorithms which have many advantages—such as generalization capabilities and the ability to discover hidden patterns for better detection of the new XML-based format Office files. As anti-viruses are capable of detecting only known malicious files and their relative variants, their ability to detect new and unknown malicious Office files is limited. There is very clearly a lack of high performance methods for the detection of malicious Microsoft Office XML-based files and the widely used docx files in particular. Moreover, given the enormous number of documents created daily, the strength of a method that will improve detection should lie in its updatability, based on efficient and frequent updates of both the detection model and commonly used anti-virus software.

Several studies in different domains have successfully applied. Active Learning techniques in order to improve detection capabilities. For example Nissim *et al.* [4] used AL methods for removing false positive in detection of unknown computer worms. A recent study, ALDROID [15] presents a system that uses AL on Android applications to enhance the updatability and detection capabilities with minimal labeled samples. Even though our methodology also use active learning for improving detection of malicious activities, it is substantially different from the other works. First of all, our system is the first to use AL methods on the structural features of DOCX documents. Second, the use of AL algorithms allow us to design a very efficient system from scalability point of view that can be embedded in the anti-virus engine improving the actual detection performance.

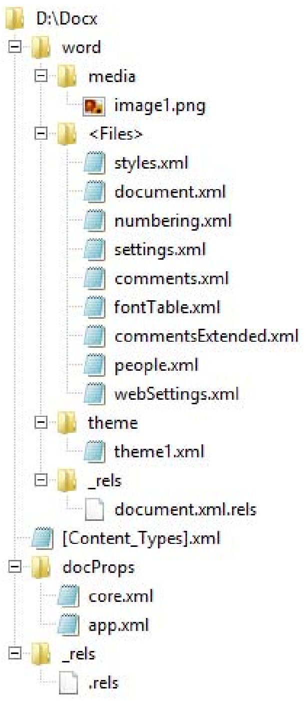
III. Office File Structure

Before we present our methods, one should understand the structure of a viable docx file. Figure 1 shows the directory tree of a sample ∗.docx file. The actual content of the file is stored in various XML files located in different folders. Each XML file holds the information of a different component. For example, 'styles.xml' file holds the styling data. 'app.xml' and 'core.xml' hold the metadata of the document, i.e., the title, document's author, the number of lines and words, etc. Based on that file structure we built our framework which is aimed at enhancing the detection of malicious docx files.

IV. Microsoft Office Files Security Threats

Now we present several of the vulnerabilities in Microsoft Office documents files. First they can contain macro which

[6]http://www.reconstructer.org/code.html
[7]http://www.sourcefire.com/
[8]http://www.aldeid.com/wiki/Officecat
[9]http://www.microsoft.com/en-us/download/details.aspx?id=2096
[10]http://www.aldeid.com/wiki/PyOLEScanner
[11]https://www.checkpoint.com/products/threatcloud-emulation-service/

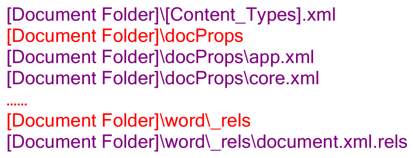


```
D:\Docx
  word
    media
      image1.png
    <Files>
      styles.xml
      document.xml
      numbering.xml
      settings.xml
      comments.xml
      fontTable.xml
      commentsExtended.xml
      people.xml
      webSettings.xml
    theme
      theme1.xml
    _rels
      document.xml.rels
  [Content_Types].xml
  docProps
    core.xml
    app.xml
  _rels
    .rels
```

Fig. 1. Example of an unzipped ∗.docx file.

is an embedded code written in a programming language known as Visual Basic for Applications (VBA). Macro code is stored in a binary file named 'vbaProject.bin' located under the 'word\' folder (in Word) in the document archive. Macro is a legitimate component; however, it can be dangerous when used for malicious purposes—posing a security risk. For example, macro can use the VBA shell command to run arbitrary commands and programs. A malicious macro code can be configured to run automatically when the file is opened using the "AutoExec" or "AutoOpen" features. Macro can download a malicious executable file from the Internet and open it, or download a malicious non-executable file such as a PDF file, and open it and take advantage of a known or unknown vulnerability in that file format. Macro can also open a malicious website in the background. Recent versions of Microsoft Office contain built-in security features that help protect the user from macro codes in general. In Microsoft Office 2003, the macro security level feature was added. In Microsoft Office 2013 all macros are disabled by default and a notification of this fact is provided. However, with a bit of social engineering, an attacker can often trick the user into enabling macros. Since the release of Microsoft Office 2007 and the new XML-based file format, only documents with file extensions ending with "m," such as ∗.docm, ∗.xlsm and ∗.pptm, can contain macro, according to Microsoft.[12] However, we found ∗.docx files that contain macro code and are labeled as malicious by some anti-virus engines. "Trusted Location" is a security mechanism based on trusted locations typically defined as a folder on a hard drive or a shared network. When a folder is designated as a trusted file source, any

[12]http://office.microsoft.com/en-001/help/introduction-to-new-file-name-extensions-HA010006935.aspx



```
[Document Folder]\[Content_Types].xml
[Document Folder]\docProps
[Document Folder]\docProps\app.xml
[Document Folder]\docProps\core.xml
......
[Document Folder]\word\_rels
[Document Folder]\word\_rels\document.xml.rels
```

Fig. 2. List of paths extracted from a sample docx file.

file that is saved in the folder is assumed to be a trusted file. When a trusted file is opened, all content in the file is enabled and active, and users are not notified about any potential risks that might be concealed within the file. These risks can include unsigned add-ins and macros, links to content on the Internet, or database connections. The default Microsoft Office security policyspecifies that unsigned macros are not allowed to be run unless they are in a trusted location.[13] Security level and trusted location security features can be bypassed using some techniques that were presented by Dechaux *et al.* [2]. In addition, Malicious Office files can contain a malicious Object Linking and Embedding (OLE)[14] object. OLE is the technology used to share data between applications, and allows, for example, an Excel spreadsheet chart to be opened within a Word document, among other functions. An OLE object is stored in a binary file named 'oleObject.bin' located under the 'word\embeddings\' folder (in Word) in the document archive. An OLE package object may contain any file or command line. If the user double-clicks on the object (located in the document), the file or the command is launched. Microsoft Office files can embed active content which may be malicious. Among the file types that can be embedded, one can find binary files, command and script files, Hypertext Markup Language files which can contain malicious JavaScript code, and other document files such as: ∗.pdf, ∗.doc, ∗.xls, and ∗.ppt which can also be malicious. Embedded malicious files can be automatically executed when the container file is opened using macro.

V. The Suggested Framework and Methods

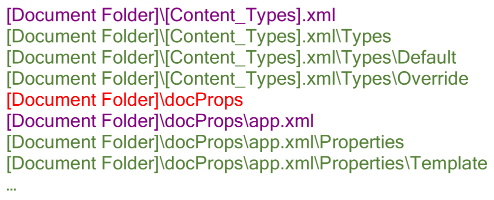
A. Structural Feature Extraction Methodology (SFEM)

We now present our new structural feature extraction methodology (SFEM) in which we use the hierarchical nature of an Office file and convert it to a list of unique paths. Figure 2 shows the list of unique paths extracted from the sample file shown in figure 1, after unzipping it. The red paths represent directories and the purple paths represent files, with 22 paths in total.

Since XML files have a hierarchical nature as well, we converted the XML files within the Office files to a list of paths by concatenating the names of the hierarchal tags within the XML, using '\'. Only the tags' names are concatenated; tags' properties and properties' values are ignored. Although tags' properties and their values contain a great deal of information that can be helpful, the integration of all the

[13]http://office.microsoft.com/en-001/access-help/security-ii-turn-off-the-message-bar-and-run-code-safely-RZ010291746.aspx?section=2

[14]http://office.microsoft.com/en-us/excel-help/create-edit-and-control-ole-objects-HP010217697.aspx



```
[Document Folder]\[Content_Types].xml
[Document Folder]\[Content_Types].xml\Types
[Document Folder]\[Content_Types].xml\Types\Default
[Document Folder]\[Content_Types].xml\Types\Override
[Document Folder]\docProps
[Document Folder]\docProps\app.xml
[Document Folder]\docProps\app.xml\Properties
[Document Folder]\docProps\app.xml\Properties\Template
...
```

Fig. 3. List of paths extracted from a sample .docx file.

properties and their values will exponentially increase the number of extracted paths and the extraction process time. Figure 3 shows the beginning of the full list of unique paths extracted from the sample file, while also extracting the paths from the XML files. The green paths represent a path of tags within an XML file. The total path count in the list is 425. The paths represent the file's properties and actions. For example, the "*\word\media\image1.png"* path means that the document contains an image, and since this is the only path under the "*\word\media\"* path, we know that this is the only media item in the file. There are a couple of paths whose presence in the file can indicate the presence of macro code, one being the "*\word\vbaData.xml"* path.

Note that all the nodes in the directory tree have been included in the paths list and not only the leaves. We do this in order to keep the generality of the higher nodes in the tree.For example, the path *'\word\vbaData.xml'* which is the ancestor of the leaf path: *'word\vbaData.xml\wne: vbaSuppData\wne:mcds\wne:mcd'*, was found to be a more powerful feature than its descendant path as it indicates the presence of macro code in the file, and not just a specific property in the vbaData.xml files. The extraction process is done statically without executing the file and is conducted quite quickly at the rate of 270ms for an average file. The most discriminative paths that are extracted from the files will be used later as features in a dataset to train the machine learning algorithm. SFEM advantages include:

- SFEM does not focus on the extraction and analysis of malicious code inside the document (as does Office-MalScanner) and therefore is a more general and robust approach for many types of attacks as it cannot be evaded by code obfuscation techniques.
- SFEM is robust against a mimicry attack in which changes were made to a malicious file to make it appear as a benign file. These changes would not affect our method significantly since the features of malicious actions still exist deep inside the structure of the file.
- SFEM is robust against a reverse mimicry attack in which changes are made to a benign file in order to make it malicious. These changes should not affect our method significantly, because the changes related to the new malicious behavior will be reflected in the file structure, and the features of malicious actions will probably be detected.
- The extraction process is done statically, without executing the file, and therefore it is more secure and requires less computational resources; as a result it can be deployed over endpoint and lightweight devices as well (e.g., smartphones).

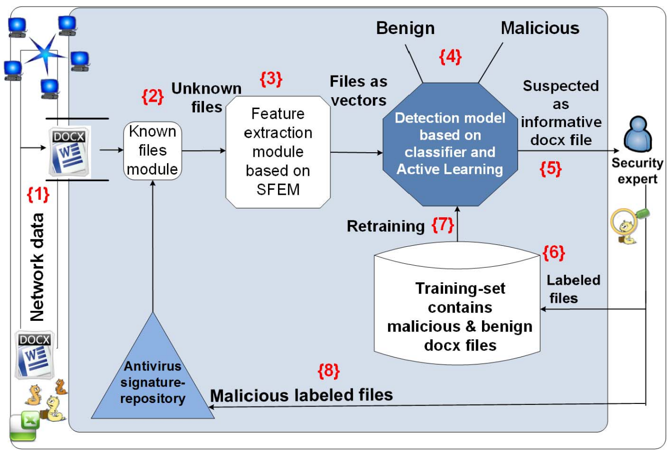

Benign
Malicious
{4}
{2}
Unknown files
{3}
Files as vectors
Suspected as informative docx file
DOCX
Known files module
Feature extraction module based on SFEM
Detection model based on classifier and Active Learning
{5}
Security expert
{1}
Retraining {7}
{6}
Network data
Labeled files
Training-set contains malicious & benign docx files
DOCX
Antivirus signature-repository
{8}
Malicious labeled files

Fig. 4. ALDOCX framework - the process of maintaining the updatability of the anti-virus tool and the detection model using AL methods pertaining to new docx files.

- The feature extraction process is relatively fast—a desired quality in online detection systems.

B. A Framework for Enhancing Detection Capabilities

Figure 4 illustrates the framework and process of detecting and acquiring new malicious docx files by maintaining the updatability of the detection model. In order to maximize the suggested framework's contribution, it should be deployed in strategic nodes (such as ISPs and gateways of large organizations) over the Internet network in an attempt to expand its exposure to as many new files as possible. Widespread deployment will result in a scenario in which almost every new file goes through the framework. If the file is informative enough or is assessed as likely to be malicious, it will be acquired for manual analysis. As Figure 4 shows, the docx files transported over the Internet are collected and scrutinized within our framework {1}. Then, the "known files module" filters all the known benign and malicious docx files {2} (according to a white list, reputation systems [3] and anti-virus signature repository). The unknown docx files are then transformed into vector form using our structural feature extraction methodology (SFEM), and these vectors represent the new files and are used for the advanced check {3}.

The remaining docx files which are unknown are then introduced to the detection model based on SVM and AL. The detection model scrutinizes the docx files and provides two values for each file: a classification decision using the SVM classification algorithm and a distance calculation from the separating hyperplane using Equation 1 {4}. A file that the AL method recognizes as informative and has indicated should be acquired is sent to an expert who manually analyzes and labels it {5}. By acquiring these informative docx files, we aim to frequently update the anti-virus software by focusing the expert's efforts on labeling docx files that are most likely to be malware or on benign docx files that are expected to improve the detection of the model. Note that informative files are defined as those that when added to the training set, improve the detection model's predictive capabilities and enrich the anti-virus signature repository. Accordingly, in our context there are two types of files that may be considered informative. The first type includes files in which the classifier has limited confidence as to their classification (the probability that they are malicious is very close to the probability that they may be benign). Acquiring them as labeled examples will probably improve the model's detection capabilities. In practical terms, these docx files will have new structural paths or special combinations of existing structural paths that represent their execution code (e.g., inside the macro code of the docx file). Therefore these files will probably lie inside the SVM margin and consequently will be acquired by the SVM-Margin strategy that selects informative files, both malicious and benign, that are a short distance from the separating hyperplane.

The second type of informative files includes those that lie deep inside the malicious side of the SVM margin and are a maximal distance from the separating hyperplane according to Equation 1. These docx files will be acquired by the Exploitation method (explanation will follow) and are also a maximal distance from the labeled files. This distance is measured by the KFF calculation that will be further explained as well. These informative files are then added to the training set {6} for updating and retraining the detection model {7}. The files that were labeled as malicious are also added to the anti-virus signature repository in order to enrich and maintain its updatability {8}. Updating the signature repository also requires an update to clients utilizing the anti-virus application. The framework integrates two main phases: training and detection/updating.

1) Training: A detection model is trained over an initial training set that includes both malicious and benign docx files. The initial performance of the detection model is evaluated after the model is tested over a stream that consists solely of unknown files that were presented to it on the first day.

2) Detection and Updating: For every unknown docx file in the stream, the detection model provides a classification, while the AL method provides a rank representing how informative the file is. The framework will consider acquiring the files based on this information. After being selected and receiving their true labels from the expert, the informative docx files are acquired by the training set, and the signature repository is updated as well, just in case the files are malicious. The detection model is retrained over the updated and extended training set which now also includes the acquired examples that are regarded as being very informative. At the end of the day, the updated model receives a new stream of unknown files on which the updated model is once again tested and from which the updated model again acquires informative files. Note that the goal is to acquire as many malicious docx files as possible since such information will maximally update the anti-virus software that protects most organizations. We employed several algorithms in order to induce detection models, including the SVM classification algorithm with the radial basis function (RBF) kernel in a supervised learning approach, and we have elected to use Lib-SVM [6]. We expect that the SVM classification algorithm will provide the best results since SVM has proven to be very efficient at enhancing the detection of malware when combined with AL methods [4], [5], [7], [8], [20], [21].

C. Selective Sampling and Active Learning Methods

Since our framework aims to provide solutions to real problems it must be based on a sophisticated, fast, and selective high-performance sampling method. We compared our proposed AL methods to other strategies, and the methods considered are described below.

1) Random Selection (Random): While random selection is obviously not an active learning method, it is at the "lower bound" of the selection methods discussed. We are unaware of an anti-virus tool that uses an active learning method for maintaining and improving its updatability. Consequently, we expect that all AL methods will perform better than a selection process based on the random acquisition of files.

2) The SVM-Simple-Margin AL Method (SVM-Margin): The SVM-Simple-Margin method [9] (referred to as SVM-Margin) is directly related to the SVM classifier. Using a kernel function, the SVM implicitly projects the training examples into a different (usually a higher dimensional) feature space denoted by F. In this space there is a set of hypotheses that are consistent with the training set, and these hypotheses create a linear separation of the training set. Among the consistent hypotheses, referred to as the version-space (*VS*), the SVM identifies the best hypothesis with the maximum margin. To achieve a situation where the VS contains the most accurate and consistent hypothesis, the SVM-Margin AL method selects examples from the pool of unlabeled examples reducing the number of hypotheses. This method is based on simple heuristics that depend on the relationship between the VS and SVM with the maximum margin. Calculating the VS is complex and impractical where large datasets are concerned and therefore the simple heuristic is used. Examples that lie closest to the separating hyperplane (inside the margin) are more likely to be informative and new to the classifier, and these examples are selected for labeling and acquisition.

This method, in contrast to ours, selects examples according to their distance from the separating hyperplane only to explore and acquire the informative files without relation to their classified labels, i.e., not specifically focusing on malware instances. The SVM-Margin AL method is very fast and can be applied to real problems; yet, as its authors indicate [9], this agility is achieved, because it is based on a rough approximation and relies on assumptions that the VS is fairly symmetric and that the hyperplane's Normal (W) is centrally placed, assumptions that have been shown to fail significantly [11]. The method may query instances in which the hyperplane does not intersect the *VS*, and therefore may not be informative. The SVM-Margin method for detecting instances of PC malware was used by Moskovitch *et al.* [10] whose preliminary results found that the method also assisted in updating the detection model but not the anti-virus application itself; however, in this study the method was only used for one day-long trial. We compared its performance to our proposed AL methods

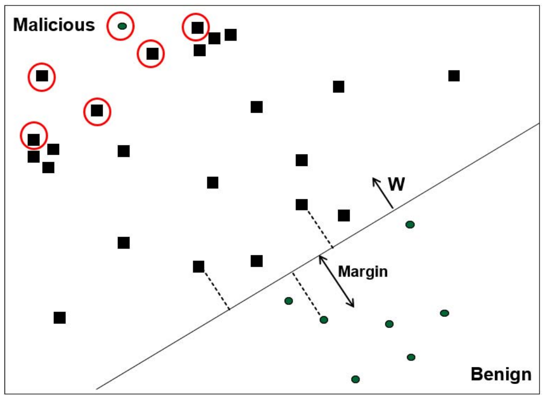


Fig. 5. The criteria by which Exploitation acquires new unknown malicious docx files. These files lie the farthest from the hyperplane and are regarded as representative files.

for a longer period, in a daily process of detection and acquisition experiments, which reflects what happens in actuality. This serves as our baseline AL method, and we expect our method to improve the new malicious docx detection and acquisition seen in SVM-Margin.

3) Exploitation (Our Proposed Active Learning Method): Our method, "Exploitation", is based on SVM classifier principles and is oriented towards selecting examples most likely to be malicious that lie furthest from the separating hyperplane. Thus, our method supports the goal of boosting the signature repository of the anti-virus software by acquiring as much new malware as possible. For every file X that is suspected of being malicious, Exploitation rates its distance from the separating hyperplane using Equation 1 based on the Normal of the separating hyperplane of the SVM classifier that serves as the detection model. As explained above, the separating hyperplane of the SVM is represented by W, which is the Normal of the separating hyperplane and is actually a linear combination of the most important examples (supporting vectors), multiplied by LaGrange multipliers (alphas) and the kernel function K that assists in achieving linear separation in higher dimensions. Accordingly, the distance in Equation 1 is simply calculated between example X and the Normal (W) presented in Equation 2.

$$Dist(X) = \left(\sum_{1}^{n} \alpha_i y_i K(x_i x)\right) \tag{1}$$

$$w = \sum_{1}^{n} \alpha_i y_i \Phi(x_i) \tag{2}$$

In Figure 5, the files that were acquired (marked with a red circle) are the files classified as malicious and have the maximum distance from the separating hyperplane. Acquiring several new malicious files that are very similar to and belong to the same virus family is considered a waste of manual analysis resources since these files will probably be detected by the same signature. Thus, acquiring one representative file for this set of new malicious files will serve the goal of efficiently updating the signature repository. In order to enhance the signature repository as much as possible, we also check the similarity between the selected files using the kernel farthest-first (KFF) method suggested by Baram *et al.* [12] which enables us to avoid acquiring examples that are quite similar. Consequently, only the representative files that are most likely malicious are selected. In cases in which the representative file is detected as malware as a result of the manual analysis, all variants that were not acquired will be detected the moment the anti-virus is updated. In cases in which these files are not actually variants of the same malware, they will be acquired the following day (after the detection model has been updated), as long as they are still most likely to be malware.

In Figure 5 it can be observed that there are sets of relatively similar files (based on their distance in the kernel space), however, only the representative files that are most likely to be malware are acquired. The SVM classifier defines the class margins using a small set of supporting vectors (i.e., docx files). While the usual goal is to improve classification by uncovering (labeling) files from the margin area, our main goal is to acquire malware in order to update the anti-virus. Contrary to SVM-Margin which explores examples that lie inside the SVM margin, Exploitation explores the "malicious side" to discover new and unknown malicious files that are essential for the frequent update of the anti-virus signature repository, a process which occasionally also results in the discovery of benign files (files which will likely become support vectors and update the classifier). Figure 5 also presents an example of a file lying far inside the malicious side that was found to be benign. The distance calculation required for each instance in this method is fast and equal to the time it takes to classify an instance in a SVM classifier, thus it is applicable for products working in real-time.

4) Combination (A Combined Active Learning Method): The "Combination" method lies between the SVM-Margin and Exploitation. On the one hand, the combination method begins by acquiring examples based on SVM-Margin criteria in order to acquire the most informative docx files (acquiring both malicious and benign files), an exploration phase which is important in order to enable the detection model to discriminate between malicious and benign docx files. On the other hand, the combination method then tries to maximally update the signature repository in an exploitation phase, drawing on the Exploitation method. This means that in the early acquisition period, during the first part of the day, SVM-Margin is more dominant compared to Exploitation. As the day progresses, Exploitation becomes predominant. However, Combination was also applied in the course of the 10-day experiment, and over a period of days, Combination will perform more Exploitation than SVM-Margin. This means that on the i$^{\text{th}}$ day there is more Exploitation than in the $(i\text{-}1)^{th}$ day. We defined and tracked several configurations over the course of several days. Regarding the relation between SVM-Margin and Exploitation, we found that a balanced division performs better than other divisions (i.e., during the first half of the study, the method will acquire more files using SVM-Margin, while during the second half of the study, Exploitation takes the leading role in the acquisition of files). In short, this method tries to take the best from both of the previous methods.

5) Comb-Ploit (A Combined Active Learning Method): Comb-Ploit is the opposite of Combination, as it starts with Exploitation in the early stages of the acquisition of new informative malicious docx files and then tries to acquire generally informative docx files. The motivation behind this is that most of the docx files are very informative and acquired during the very early stages. The method then changes its strategy and tries to improve its knowledge store by also acquiring benign docx files.

VI. Evaluation

A. Dataset Collection

In order to evaluate our proposed framework and methods, we created as large and representative dataset (as possible) of malicious and benign Microsoft Word files (∗.docx). We acquired a total of 16,811 files, including 327 malicious and 16,484 benign files, from the four sources presented in Table I. All the files were assured to be labeled correctly as malicious or benign using *VirusTotal*[15] service. As is known, VirusTotal is a malware repository that offers a free service for malware detection using various anti-virus engines. Users from all over the world upload benign and suspicious files to VirusTotal for examination. The malicious files in our collection were gathered from VirusTotal during 2015, and were uploaded to VirusTotal between 2010 and 2015. In our experiment we only considered malicious files that were detected by at least five anti-virus engines in order to ensure a high quality dataset that was free from mislabeled instances. Note that we also provide the distribution of the threats in our malicious collection in Figure 6 which shows that our malicious collection contains a wide range of threats and a variety of virus families which should satisfactorily represent the population of malicious MS ∗.docx documents.

In Figure 6 we can see the malware families' distribution of our malicious docx files within the dataset. As can be seen, our malicious collection contains a wide range of threats and a variety of virus families, which should satisfactorily represent the population of malicious MS ∗.docx documents.

B. Dataset Creation

We developed a feature extractor based on our structural feature extraction methodology (SFEM) in order to extract and analyze the features from all the files in the dataset. The feature extraction process resulted in a vocabulary of 134,854 unique features extracted from both benign and malicious files. For each feature we counted its occurrence among benign files and malicious files for statistics and feature election purposes as multiple occurrences of a feature in a specific document file is counted once. In order to select the most prominent features from the list of 134,845 features, we used a feature selection method based on *Information Gain*. *Information Gain* gives a higher grade to features that contribute primarily to discrimination between malicious and benign classes. We sorted the features according to their *Information Gain* grade and were left with a list of the 5,000 most prominent features,

[15]https://www.virustotal.com/

TABLE I
THE SOURCES OF OUR DATASET

Dataset Source	**Year**	**Malicious files**	**Benign files**
VirusTotal repository	2010-2015	327	15,517
Contagio[16] collection	2013	----	100
Ben-Gurion Uni' (Random)	2010-2015	----	867
Total		**327**	**16,484**

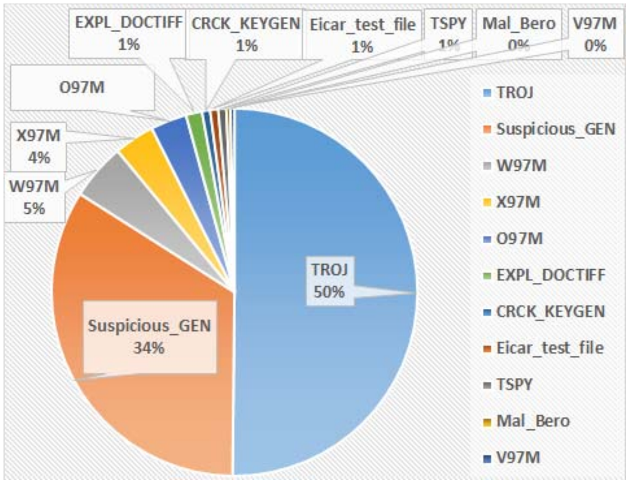


Fig. 6. Segmentation of the malicious docx files into virus-families according to the Trend-Micro anti-virus.

which we called the "Top 5000." Using the ranked list of the 5,000 most prominent features and the collection of 16,811 ∗.docx files (benign and malicious), we created the dataset for our experiments which contained 16,811 records and 5,001 fields. Each record in the dataset represents a file in the collection. Fields 1 to 5,000 represent the features, where the first feature represents the most prominent one and the 5,000th field represents the least prominent. In order to represent the files as records we used the *Boolean* representation to indicate the presence (1) or absence (0) of a feature within the file. Field 5,001 represents the class of the file, "Malicious" if the record represents a malicious docx file and "Benign" if the record represents a benign docx file. In order to check how the number of features affects the detection rates, we also took into consideration subsets of prominent features and created 8 datasets using the 10, 40, 80, 100, 300, 500, 800 and 1000 top features. The top features in which the best detection rates are achieved, will be the features that will be used for further experiments.

Note that the paths that are extracted from the document represent the document's general structure. This means that if the document contains media files (e.g., ∗.jpg) or code files (e.g., vbaproject.bin), this is reflected in the paths. Moreover, the paths also represent information regarding the content and configuration of the document, since they are extracted from the ∗.xml file which contains the configurations and content of the document. In particular, SFEM

[16]http://contagiodump.blogspot.co.il/

TABLE II
THE TOP 11 PROMINENT FEATURES AND THEIR CATEGORIES

Feature (structural path)	Category
f1: *word\vbaData.xml*	*Macro*
f2: *word\vbaData.xml\wne:vbaSuppData*	*Macro*
f3: *word_rels\vbaProject.bin.rels*	*Macro*
f4: *word_rels\vbaProject.bin.rels\Relationships*	*Macro*
f5: *word_rels\vbaProject.bin.rels\Relationships\Relationship*	*Macro*
f6: *word\vbaProject.bin*	*Macro*
f7: *word\vbaData.xml\wne:vbaSuppData\wne:mcds*	*Macro*
f8: *word\vbaData.xml\wne:vbaSuppData\wne:mcds\wne:mcd*	*Macro*
f9: *word\embeddings*	*Embedded*
f10: *word\embeddings\oleObject1.bin*	*OLE*
f11: *word\media\image1.emf*	*EMF*

TABLE III
PERCENTAGE OF OCCURRENCE OF TOP 11 FEATURES AMONG BENIGN AND MALICIOUS FILES

#	Category	Occurrence Percentage in benign files	Occurrence Percentage in malicious files
f1	*Macro*	0.13%	44.04%
f2	*Macro*	0.13%	44.04%
f3	*Macro*	0.13%	44.04%
f4	*Macro*	0.13%	44.04%
f5	*Macro*	0.13%	44.04%
f6	*Macro*	0.16%	44.34%
f7	*Macro*	0.12%	40.98%
f8	*Macro*	0.12%	40.98%
f9	*Embedded*	3.14%	59.94%
f10	*OLE*	2.43%	44.04%
f11	*EMF*	1.90%	40.37%

also extracts features that are strongly indicative of, and related to, the attack techniques used in malicious MS Office files (presented in section 3): (1) malicious macro code via VBA code embedded in the docx file, and (2) malicious Object Linking and Embedding (OLE) objects (presented in Table II). Table II presents the 11 most prominent features. Features f1-f8 are related to the existence of macro code in the document and its activation, Feature f9 is related to the existence of embedded files, feature f10 is related to OLE objects in the document. Feature f11 signifies the presence of an $*$.emf image in the document.

Table III shows the percentage of occurrences of the first 11 prominent features, f1-f11, within benign and malicious files. These percentages shed some light and provide a basis to better understand the dataset's composition regarding the selected features. Looking at features f1-f8, we can see that about 44% of the malicious files contain macro code, whereas among the benign files only 0.13% to 0.16% contain macro. Additionally, almost 60% of the malicious files contain an embedded file compared to only 3.14% of the benign files. However these numbers and features are not enough to achieve sufficiently high detection rates, and therefore we leverage the features and induce a detection model by conducting several experiments using different subsets of features and different classifiers in order to identify the set of the most discriminative features and the best classifier.

The most popular attacks through docx files are launched via macro, file embedding, and OLE categories, and it is significant that the most prominent features belong to these categories. Note also that the eleventh feature is indicative of the presence of an EMF (Enhanced Meta File). We checked the appearance of the EMF feature (f11) in our dataset and came to the conclusion that it appears in many families and not just in many variants of one family. Therefore, the EMF's existence is actually a strong, additional indication of maliciousness.

C. Experimental Design

Our experimental design aims at providing clear and practical answers to the following research questions:

1. Is it possible to detect new and unknown malicious docx files based on the structural feature extraction methodology and machine learning algorithms? What is the optimal configuration yielding the best detection results in terms of number of features (top) and classifier?
2. On a daily basis, is it possible to improve both the detection model's performance and the anti-virus detection capabilities by enlarging the signature repository with new unknown malicious docx files using AL methods and the new structural feature extraction methodology?
3. Is it possible to leverage and utilize an organization's large collection of unlabeled docx files to create an accurate detection model and efficiently update the anti-virus software through the acquisition of only a minimal, yet informative set of docx files while also reducing the labeling efforts of the security experts?

For each one of these questions, we designed a comprehensive and specific experiment. Thus here we present an experiment which pertains to each research question, respectively.

1) Detection Evaluation and Optimal Configuration: In order to answer the first research question and evaluate the detection capabilities of machine learning algorithms based on the new structural feature extraction methodology, we used the dataset we collected and built, consisting of 16,811 docx files with 1.9% malicious files (16,484 benign, 327 malicious). We evaluated the performance taking into account different classifiers: J48, Random-Forest, LogitBoost, Logistic Regression and Support Vector Machines (SVM). We also took into consideration different tops in order to find the optimal and most discriminative set of structural features for our detection purposes. The tops were: 10, 40, 80, 100, 300, 500, 800 and 1000 features. Five different classifiers with eight different tops resulted in forty different experiments for each. We applied the standard ten-fold XV over this dataset, which means dividing the dataset into ten-equal folds with randomly selected docx files. We performed training on nine folds and testing on the last remaining fold, repeating this process ten times, and finally averaging the results. Thus, training was performed on 90% ($\sim$15,129 docx files) of the dataset, and testing was performed on the remaining 10% ($\sim$1,682 docx files). Note that both the training and the test sets contained 1.9% malicious files in order to adequately represent reality as closely as possible. By doing so, our results are not biased regarding incorrect and unreasonable malicious docx files within the test set. Having a test set made up of 50% malicious files definitely creates a bias in the results and distorts the conclusions and insights of the research. In contrast, we attempted to portray a realistic sample and avoid a biased result. In order to have a practical impact and emphasize the contribution of our framework, we also compared the performance of our detection models with popular anti-virus software used by organizations.

2) Updatability and Detection Enhancement Evaluation: The objective of this experiment was to evaluate and compare the performance of our new AL methods to the existing selection method, SVM-Margin and provide answers to the second research question based on the following two tasks:

- acquiring as many new, unknown malicious docx files as possible on a daily basis in order to efficiently enrich the signature repository of the anti-virus
- updating the predictive capabilities of the detection model that serves as the knowledge store of AL methods and improve its ability to efficiently identify the most informative new malicious docx files

Over a ten day period, we compared docx file acquisition based on AL methods to random selection based on the performance of the detection model. In our acquisition experiments we used 16,811 docx files (16,484 benign, 327 malicious) from our repository and created ten randomly selected datasets with each dataset containing ten subsets of 1,600 files which represent each day's stream of new files. The 811 remaining files were used by the initial training set to induce the initial model. Note that each day's stream contained 1,600 docx files. At first, we induced the initial model by training it on the 811 known docx (795 benign, 16 malicious) files. We then tested it on the first day's stream. Next, from this same stream, the selective sampling method selected the most informative docx files according to that method's criteria. The informative files were sent to an expert who thoroughly inspected and labeled them. The files were later acquired by the training set which was enriched with an additional X number of new informative files. When a file was found to be malicious, it was immediately used to update the signature repository of the anti-virus, and an update was also distributed to clients. The process was repeated over the next nine days. The performance of the detection model was averaged for ten runs over the ten different datasets that were created. Each selective sampling method was checked separately on five different acts of file acquisition (each consisting of a different number of docx files). This means that for each act of acquisition, the methods were restricted to acquiring a number of files equal to the amounts that followed, denoted as X: 10 files, 20 files 50 files 100 files and 250. We also considered the acquisition of all the files in the daily stream (referred to as the ALL method), which represents an ideal, but not a feasible way, of acquiring all the new files and more specifically, all the malicious docx files. The ALL method was considered in order to compare the effectiveness of our methods against the maximal malicious docx file acquisition. The experiment's steps are as follows:

1. Inducing the initial detection model from the initial available training set, i.e., the training set available up to day d (the initial training set includes 811 docx files).
2. Evaluating the detection model on the stream of day $(d + 1)$ to measure its initial performance.
3. Introducing the stream of day $(d + 1)$ to the selective sampling method which chooses the X most informative files according to its criteria and sends them to the expert for manual analysis and labeling.
4. Acquiring the informative files and adding them to the training set, as well as using their extracted signature to update the anti-virus signature repository.
5. Inducing an updated detection model from the updated training-set and applying the updated model on the stream of the next day $(d + 2)$.

This process repeats itself on our dataset from the first day until the tenth day.

3) Leveraging and Utilizing Documents within the Organization for Detection of Malicious Docx Files: The objective of the third experiment which compares the performance of our new AL methods to the existing selection method, SVM-Margin, was to measure and evaluate our framework's contribution when applying it inside an organization for both enhancing the capabilities of the detection model and anti-virus, as well as reducing the labeling efforts of the security expert when labeling the most informative docx files. In the second experiment we wanted to evaluate the performance and capability of our suggested framework to efficiently update both the anti-virus and the detection model. The evaluation was done through the daily acquisition of new docx files over the course of ten days. This means that each day the framework encounters new docx files that haven't yet been seen by either the anti-virus or the detection model. These new files (arriving on day d) are (intentionally) not used by the framework for acquisition on the d + 1 day since the framework deals with the new files arriving on the d + 1 day, however the day d's files may be contributory and informative over the course of the following days. This experiment aims to evaluate the framework with respect to this scenario, incorporating the challenging task of including historical docx files which organizations already have within their networks and leveraging and utilizing these files in order to enhance the detection's model performance in the very early stages of acquisition. During the acquisition trials that ended by acquiring every docx file in the pool of unlabeled docx files, we compared the acquisition of docx files based on AL methods to random selection based on the performance of the classification model. In our acquisition experiments, we used all 16,811 docx files (16,484 benign, 327 malicious) in our repository and created ten randomly selected datasets with each dataset containing three elements: an initial set of 1,811 docx files randomly selected that were used to induce the initial detection model; a test set of 1,000 docx files upon which the detection model was tested and evaluated after every trial in which it was updated; and a pool of the remaining 14,000 unlabeled and unknown docx files, from which the framework and the selective sampling method selected the most informative and most likely malicious docx files according to that method's criteria. The informative docx files were sent to a security expert who inspected and labeled them. The docx files were later acquired by the training set that was enriched with an additional X new informative docx files. When a file was found to be malicious, it was immediately used to update the signature repository of the anti-virus, and an update was also distributed to clients. The process was repeated over the next trials until the entire pool was acquired. The performance of the detection model was averaged for ten runs over the ten different datasets that were created. Each selective sampling method was checked separately on the five different acts of docx files acquisition (each consisting of a different number of docx files). This means that for each act of acquisition, the methods were restricted to acquiring a number of files equal to the amount that followed, denoted as X: 10 files, 20 files, 50 files 100 and 250 files. Out of the total 16,811 files, we used 1,811, a relatively small number of docx files (∼1774 benign, ∼37 malicious) as an initial set for the following reasons. First, as labeling files can become costly due to the need for a security expert to inspect them, we tried to reduce the need for extensive labeling efforts. Secondly, we wanted to induce an initial detection model that had a medium performance (not too low and not too high) that could be leveraged and improved through intelligent acquisition of docx files from a large pool of docx files. The experiment's steps are as follows:

1. Inducing the initial detection model from the initial available training set (the initial training set includes 1,811 docx files).
2. Evaluating the detection model on the test set of 1,000 docx files to measure its initial performance.
3. Introducing the pool of unknown and unlabeled docx files to the selective sampling method which chooses the X most informative docx files and sends them to the security expert for inspection and labeling.
4. Acquiring these labeled informative docx files, removing them from the pool and adding them to the training set, as well as using their extracted signature to update the anti-virus signature repository (in the case that they were found to be malicious).
5. Inducing an updated detection model from the updated training set and applying the updated model on the pool (which now contains fewer docx files).

This process is repeated on our dataset from the first trial until the entire pool is acquired.

VII. Results

A. Detection Evaluation and Optimal Configuration

We now present the results of the detection and configuration experiment for five different classifiers trained on eight different tops, based on the SFEM of the docx files. As the dataset is imbalanced and consists of only 1.9% malicious files—in order to reflect reality as closely as possible—the TPR measure is the most accurate measure for evaluating the detection capabilities achieved by each configuration. As can be seen in figure 7, the tops for which most of the classifiers

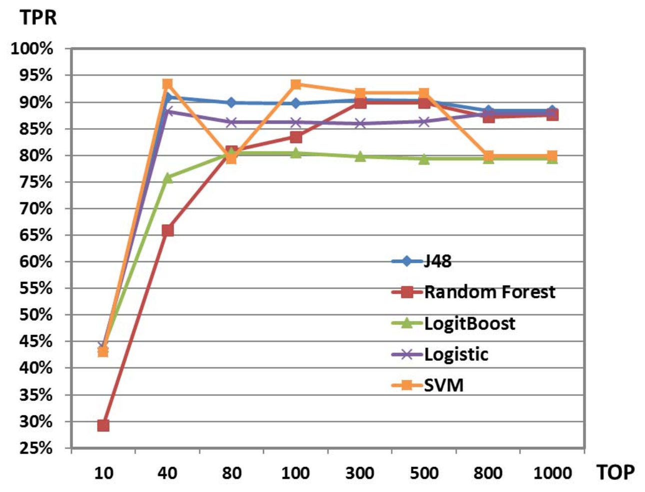


Fig. 7. The TPR rates for five different classifiers in detecting malicious docx files over the eight different tops of the features.

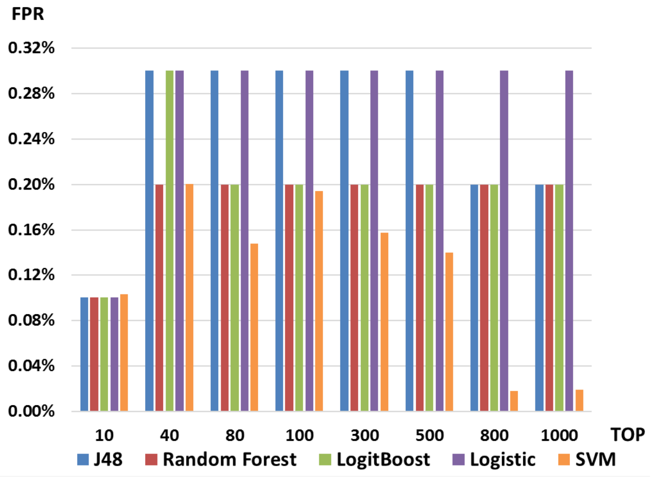


Fig. 8. The FPR rates for five different classifiers in detecting malicious docx files over the eight different tops of the features.

had the highest TPR levels are 40, 100, 300 and 500 features. Specifically, SVM outperformed the other classifiers on these tops and achieved the highest TPR rates of 93.48% and 93.34%, respectively for top 40 and top 100 features. J48 was the second best performer and its TPR rates were around 90% for most of the tops. In top 10, all the classifiers had very low detection rates of no more the 45%, which indicates that these ten features that represent the macro, file embedding and OLE were not informative enough to distinguish between the malicious and benign docx files. This fact has major implications regarding the resilience and robustness of our framework towards different docx malwares as will be discussed later in this article.

As high TPR rates represent strong detection capabilities, these detection capabilities should also have low FPR rates so that any false alarms raised in response to benign files will be minimal—and this is what also strengthens the reliability of the framework in detecting malicious docx files. Figure 8 shows that in general, all the tops and classifiers had

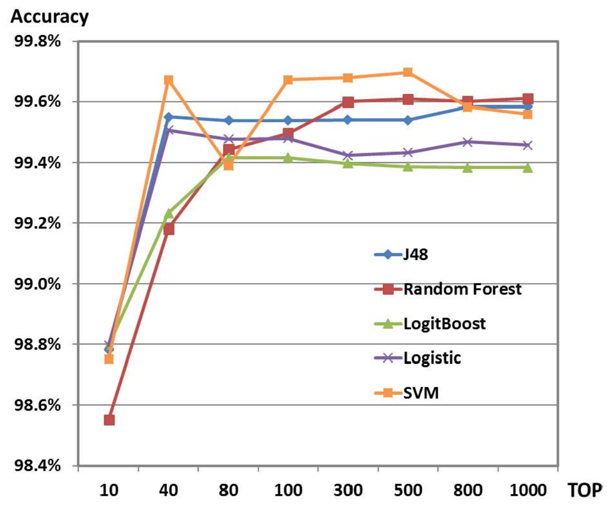


Fig. 9. Accuracy rates for five different classifiers in detecting malicious docx files over the eight different tops of the features.

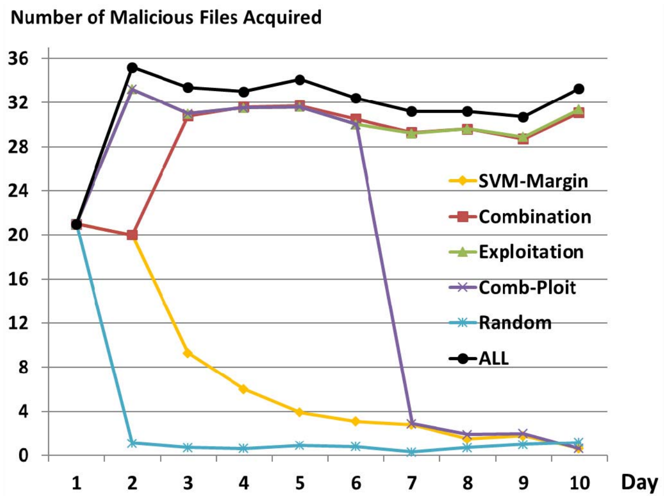


Fig. 10. The number of malicious docx files acquired by the framework for different methods with acquisition of 50 files daily.

a very low FPR rate, ranging between 0.019% and 0.02%, which was very encouraging and supports the effectiveness of our structural features and the information provided by them. The lowest FPR levels were achieved by SVM using the larger number of features such as the 800 and 1,000 features.. As 40 and 100 were the best tops, providing the highest TPR rates with the SVM classifier, and their FRP levels were 0.02% and 0.019%, respectively.

Taking into consideration the general accuracy of the framework in correctly classifying malicious and benign docx files, almost a 100% accuracy rate is shown in Figure 9. One should note that it is an easy task to achieve an accuracy rate of 98.1% since the data is imbalanced, and contains 98.1% benign files. However, each percentage point above 98.1% is more challenging. As shown in the above figure, all the classifiers' general accuracy is of at least 98.5%, while the SVM with tops 40, 100, 300 had accuracy of 99.67% and 99.69% with top 500.

After evaluating the different classifiers and tops using these three important measures, we can conclude that the configuration that provides the best detection capabilities is the SVM classifier trained on top 100 features: TPR of 93.34%, FPR of 0.19% and accuracy of 99.67%. Although using top 40 SVM achieved a 0.14% higher TPR rate, we will use top 100 features because this had a lower FPR (0.1% lower), since a false alarm on benign file (FP) is a very expensive event, which should be minimized when dealing with detection of malicious files, particularly within organizations.

B. Updatability and Detection Enhancement Evaluation

We rigorously evaluated the efficiency and effectiveness of our framework, comparing five selective sampling methods: (1) a well-known existing AL method, termed SVM-Simple-Margin (SVM-Margin) based on [9]; our proposed methods, (2) Exploitation, (3) Combination, (4) Comb-Ploit, and (5) random-selection (Random) as a "lower bound." Each method was checked for all five acquisition amounts in which the results were the mean of ten different folds. Due to space limitations, we depicted the results of the most representative acquisition amount of 50 docx files—which is a small amount more than the average number of malicious files (33) found in the daily stream. The 50 docx files sent to the security expert for inspection were a reasonable number that could be dealt with within one day and a meaningful amount that could contribute to the efficient and frequent update of the detection model and the anti-virus.

We now present the results of the core measure in this study, the number of new, unknown malicious files that were discovered and finally acquired into the training set and signature repository of the anti-virus software. As explained above, each day the framework dealt with 1,600 new docx files, consisting of about 33 new, unknown malicious docx files (1.9%). Statistically, the more files that are selected daily, the more malicious files will be acquired daily. Yet, using AL methods, we tried to improve the number of malicious files acquired by means of existing solutions. More specifically, using our methods (Exploitation, Combination and Comb-Ploit) we also sought to improve the number of files acquired by the SVM-Margin. Figure 10 presents the number of malicious docx files obtained by acquiring the 50 files daily, by each of the five methods during the course of the ten day experiment. Exploitation and Combination outperformed the other selection methods. Exploitation and Comb-Ploit were the only methods that showed an increasing trend from the first day to the second day, while Combination had a decrease in the first day and then had the same performance as Exploitation over the following days. During the course of the ten days, SVM-Margin and Random selection had a decreasing trend in the number of malicious docx files acquired and showed the poorest performance in updating the detection model and anti-virus with new malicious docx files. Comb-Ploit was designed to act 50% of the time like Exploitation and during the remaining 50% of the time like SVM-Margin, and its acquisition performance is depicted accordingly. Exploitation succeeded in acquiring the maximal number of malwares from the 50 files acquired daily and outperformed all the other methods.

On the first day, the number of new malicious docx files was 21 since the initial detection model was trained on an initial set of 811 labeled docx files that consisted of only 21 malwares (1.9%). We decided on 811 files from which the initial detection model would be induced in order to have a stable detection model with quite a low detection performance from the start (42.13% TPR on the first day) that can be improved through our active learning based framework.

On the tenth day, using Combination and Exploitation, 94% of the acquired files were malicious (31 out of 33); using SVM-Margin, only 3% of the acquired files were malicious (1 file out of 33—less than Random). This presents a significant improvement of 91% in unknown docx malware acquisition. Note that on the tenth day, using Random, only 6% of the acquired docx files were malicious (2 out of 33). This is far less than the malware acquisition rates achieved by both Combination and Exploitation. It can be seen that the performance of our methods was much closer to the ALL line which represents the maximum malicious docx files that can be acquired each day. Therefore, in comparing the acquisition graph lines of Combination and Exploitation to the ALL graph line, the trend is quite clear from the third day: each day, Combination and Exploitation maintained the same high acquisition rates of 93-94% of malicious files despite the fact that every day there were new, unknown files—a finding that demonstrates the impact of updating the detection model with new informative files, as this results in identifying new malwares for enriching the signature repository of the anti-virus. Moreover, the acquired malwares are expected to be of higher quality in terms of their contribution to the detection model, as well as the signature repository since they are different. Based on our observations, from the second day, the random selection trend remained constant; with no improvement in acquisition capabilities over the ten days. SVM-Margin AL method showed a decrease in the number of malwares acquired from the first day. This phenomenon can be explained by looking at the ways the methods work. The SVM-Margin acquires examples about which the detection model is less confident. Consequently, they are considered to be more informative but not necessarily malicious. As was explained previously, SVM-Margin selects new informative docx files inside the margin of the SVM. Over time and with the improvement of the detection model regarding more malicious files, it seems that the malicious files are less informative (due to the fact that malware writers frequently try to use upgraded variants of previous malwares). Since these new malwares might not lie inside the margin, SVM-Margin may actually be acquiring informative benign files rather than malicious ones. However, our methods of Combination and Exploitation are more oriented toward acquiring the most informative files and most likely malware by obtaining informative docx files from the malicious side of the SVM margin. As a result, an increasing number of new malwares are acquired; in addition, if an acquired benign file lies deep within the malicious side, it is still informative and can be used for learning purposes and to improve the next day's detection capabilities. We have shown here that our AL methods outperformed the SVM-Margin AL method and improved the capabilities for acquiring new docx malwares and enriching the signature repository of the anti-virus software. In addition, as is shown in Figure 11, our methods also maintain the predictive performance performance of the detection model that serves as the knowledge store of the acquisition process.

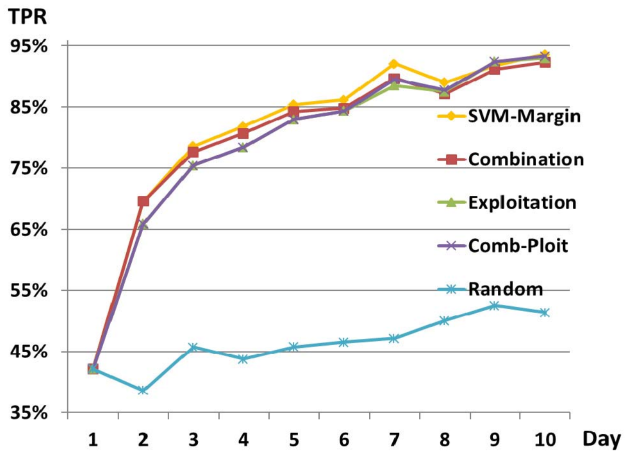


Fig. 11. The TPR of the framework over the 10 days for different methods through the acquisition of 50 docx files daily.

Figure 11 presents the TPR levels and their trends during the 10-day course of study. SVM-Margin outperformed other selection methods in the TPR measure, while our AL methods, Combination, Exploitation and Comb-Ploit, came very close to SVM-Margin (SVM-Margin achieved 0.4% better TPR rates than Comb-Ploit, 0.5% better than Exploitation and 1.3% better than Combination) and performed much better than Random. In addition, the performance of the detection model improves as more files are acquired daily, so that on the tenth day of the experiment, the results indicate that by only acquiring a small and well selected set of informative files (50 docx files out of 1600 are 3% of the stream), the detection model can achieve TPR levels (93.6% with SVM-Margin, 93.2% with Comb-Ploit, 93% with Exploitation and 92.3% with Combination) that are quite close to those achieved by acquiring the whole stream (94.4%). These results will probably have economic implications and demonstrate the efficiency of the framework in maintaining and improving the updatability of the detection model and ultimately of the anti-virus software. These factors demonstrate the benefits obtained by performing this process on a daily basis.

To better demonstrate the contribution of AL methods within ALDOCX, we will now compare the TPR rates with and without AL methods. The TPR rate of 93.6% was achieved here through the AL process using only 1,311 docx files (811 initial set + 500 acquired after ten days) out of the total 16,811, which is 7.7%. Whereas, in the previous experiment of detection evaluation, the best TPR rate was 93.34% which was achieved using the SVM classifier with top 100 when trained with over 90% of the dataset. This led to a reduction of 91.4% in labeling efforts of unknown docx files

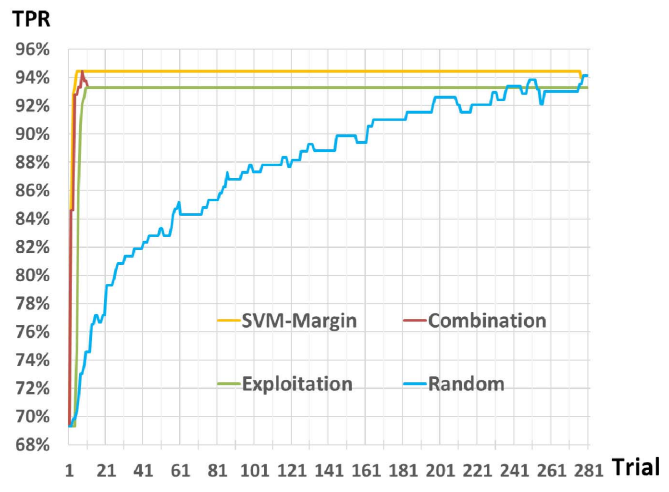


Fig. 12. The TPR of the framework over the 280 trials of acquisition for different methods through the acquisition of 50 docx files daily from the organization's collection.

and even induced a more accurate detection model of 93.6% TPR compared to 93.34% TPR.

The results of the predictive capabilities demonstrated by the TPR and FPR measures indicate that our methods, Exploitation and Combination, performed as well as the SVM-Margin method with regard to predictive capabilities (TPR and FPR) but, as can be seen in Figure 10, much better than the SVM-Margin in acquiring a large number of new docx malwares daily and enriching the signature repository of the anti-virus.

C. Leveraging and Utilizing Documents within the Organization for Detection of Malicious Docx Files

Comparing AL methods to the random selection (passive learning) we can see in Figure 12 that the SVM-Margin achieved the highest TPR of 94.4% after only five trials. This indicates that only 2,011 informative docx well selected files (14% of the pool of unlabeled docx files) were required to induce this accurate detection model. Both Exploitation and Combination achieved a high and stable TPR rate of 93.2% TPR after ten trials which means only 2,511 informative and well selected docx files (18% of the pool of unlabeled docx files). It took the Random selection 237 trials which is 13,661 labeled files, representing 97.5% of the docx files in the pool. Therefore we demonstrate a reduction of 81% in labeling efforts when comparing Random selection to our AL methods and still maintain low FPR rates of 0.18%. The previous experiment demonstrated that Comb-Ploit has no advantage over other selection methods, and therefore we did not evaluate it in this third experiment.

In Figure 13 we can see that during very early stages, both of our AL methods acquired 304 malicious docx files out of the 312 present in the pool (97.4% of the malicious docx files), and updated the detection model and anti-virus signature repository. This was achieved in the 12th trial, while it took the Random method 270 trials to acquire the same number of malicious docx files, and the SVM-Margin required the entire pool, or 280 trials. Ultimately, ALDOCX provided a reduction of 95.5% and 95.7% respectfully with Random and SVM-Margin in the labeling efforts and updated both the detection model and the anti-virus software with new informative docx files – benign and especially malicious.

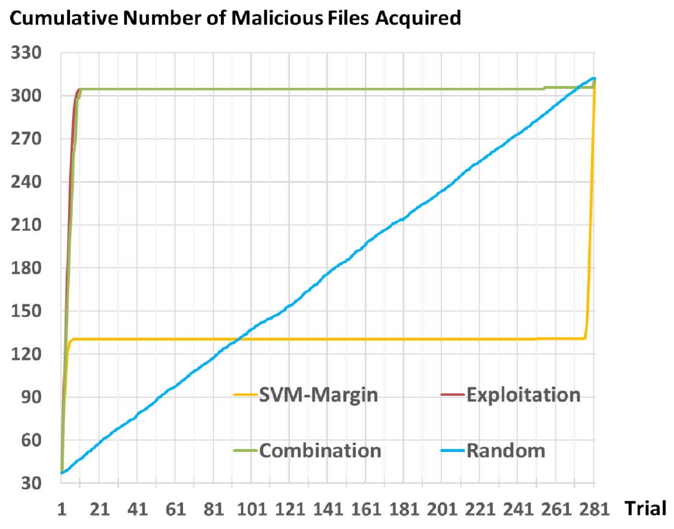


Fig. 13. The cumulative number of malicious docx files acquired by the framework for different methods with acquisition of 50 files daily from the organization's collection.

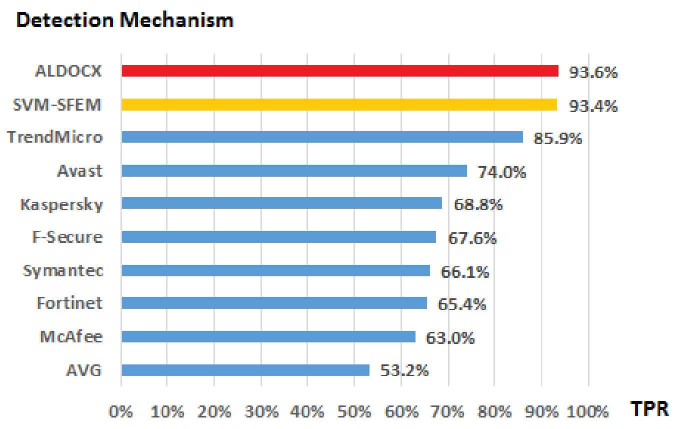


Fig. 14. The TPR of the framework against 10 best anti-viruses commonly used within organizations.

After presenting the results of the framework in three different experiments, including its detection performance and updatability towards malicious docx files, we now compare the detection rate of our methods with the best 10 anti-viruses commonly used by organizations. From Figure 14, it can be understood that the most accurate anti-virus, *TrendMicro*, had a detection rate of 85.9%, while our methods outperformed all the anti-viruses in the task of detecting new malicious docx files. Using the SVM classifier with 100 structural features (SFEM), we achieved 93.4% which required using 90% of the dataset for training (due to the 10XV settings). Using the full ALDOCX framework, including the AL methods and its enhancement process, we even improved the performance of SVM-SFEM, achieving a 94.4% TPR using only 14% of labeled data (2,011 docx files out of 14,000), which means a reduction of 84.4% in labeling efforts.

VIII. Discussion and Conclusion

We presented ALDOCX, a framework for enhancing the detection of unknown malicious Microsoft Word documents using designated active learning methods. ALDOCX is mainly based on machine learning algorithms trained with our new Structural Feature Extraction Methodology (SFEM), which is static, fast and robust against the different attacks launched through docx files. We evaluated our framework through three comprehensive experiments using the dataset we built. Our dataset was large, updated and representative, consisting of 16,811 docx files (malicious and benign). In the first experiment, we found that among five classification algorithms and eight different alternatives of choosing the top features, the configuration that yielded the best results was the SVM classifier trained on the top 100 structural features, which achieved a TPR of 93.34%, FPR of 0.19% and an accuracy of 99.67%. If the number of features plays a significant role, then top 40 could also be used for achieving nearly the same results. The number of features (top 40, top 100) showed that the detection of malicious docx files with high TPR rates requires the consideration of more than just the top ten "trivial" features (macro, embedding, OLE) and must include features that are extracted from deep within the structure of the docx file. Note that when an attacker attempts to create a new malicious document or even tries to turn a benign document into a malicious one, he/she actually changes the document's structure. Some structural paths are associated with malicious documents and others with benign documents, and these structural features allow the machine learning classifier to efficiently discriminate between malicious and benign documents. Ultimately, when an attacker tries to turn a benign docx document into a malicious one, unique structural paths are automatically generated by the MS Office editing software. The complexity of modifying the paths that may indicate that the file is malicious is similar to the complexity involved in hiding steganography in a picture. The attacker cannot simply eliminate incriminating paths that are generated when he/she embeds malicious components in a docx file. The robustness of the method is demonstrated further as follows. Both the TPR and Accuracy rates presented in Figures 7 and 9 (respectively) suggest that including only the 10 top prominent features (which are directly related to the attacks) is not sufficient to achieve a high detection rate, since when using only the 10 most prominent features, the maximal TPR was 45% and the maximal accuracy was 98.8%, (not a high level of accuracy when the dataset is imbalanced and consists of only 1.9% malicious files). However, when considering additional features (40 or more), SFEM also includes features which are not directly related to attacks – features that are associated with other structural elements in the docx file – therefore the maximal results were much higher and were 94%, 0.2%, and 99.7% TPR, FPR, and accuracy, respectively. These features are not trivial and cannot be easily changed or manually hidden, since they are created automatically when a file is edited by MS Office software; changing them manually will likely create incompatibility within the docx file. Our results show the high TPR and accuracy rates, along with the low FPR rates achieved when using such features in addition to those related to the attacks.

These non-trivial features are hard for an attacker to learn and cannot be easily evaded. The results in the first detection experiment show that the ALDOCX framework can be integrated into Microsoft Office tools or can be deployed on endpoints in order to detect malicious docx files.

Using our designated active learning methods within ALDOCX, we showed that we can efficiently update anti-virus software with unknown malicious docx files. With our updated classifier, we can detect better new malicious docx files that can be utilized for sustaining anti-virus software. Both the anti-virus and the detection model (classifier) must be updated with new and labeled docx files. Such labeling is done manually by human experts, thus the goal of the active learning is to focus expert efforts on labeling files that are more likely to be malicious docx or on docx files that might add new information about benign files. Our proposed framework is based on our active learning methods (Exploitation, Combination, and Comb-Ploit), specially designed for acquiring unknown malware. The framework seeks to acquire the most informative docx files, benign and malicious, in order to improve classifier performance, enabling it to frequently discover and enrich the signature repository of anti-virus software with new unknown malware.

In general, four of the AL methods performed very well in updating the detection model, with two of our methods, Combination and Exploitation, outperforming SVM-Margin in the main goal of the study which is acquisition and detection of new unknown malicious docx files. The evaluation of the classifier before and after the daily acquisition showed an improvement in the detection rate, and subsequently more new malwares were acquired. On the 10th day, Combination and Exploitation acquired more than ten times more malicious docs files (31) than the number acquired by SVM-Margin (3 docx files) and more than five times more malicious docx than those acquired by the Random method (6 malicious docx files). While our Combination and Exploitation methods showed a stable trend and almost perfect acquisition rates in the number of malicious docx files in the course of the 10 days, SVM-Margin and Random showed a steep decrease and poor performance along the 10 days. Therefore, our framework was found to be effective in updating the anti-virus software by acquiring the maximum number of malicious PDF files. It also maintains a well updated model that is aimed at daily detection of new and unknown malicious docx files. By acquiring only 50 docx files out of the new 1600 docx files presented to the framework each day, we showed that our Active Learning methods can provide a reduction of 91.4% in labeling efforts of unknown docx files (we needed only 7.7% of the docx files to be labeled) and even inducing a more accurate detection model with 93.6% TPR compared to the results achieved in the first experiment of 93.34% TPR. The 93.34% TPR rate on the first experiment was achieved without active learning using 90% of the docx files to be labeled. These results show that the ALDOCX framework can be deployed on strategic nodes of the Internet network in order to actively identify, select and acquire the most informative and most likely malicious docx files, and thus provide an efficient, frequent, and valuable update for anti-virus software that is widely used by organizations. The results of the third experiment show that our framework is capable of utilizing the vast number of docx files within organizations for efficiently and effectively inducing an accurate detection model. Additionally, it shows that our framework is capable of updating the anti-virus software using only a minimal set of well selected, informative docx files and significantly reducing the labeling efforts required by security experts. Moreover, the fact that in this historically-based experiment, we achieved higher TPR rates (94.44%) than the non-historical (93.6%) AL experiment, emphasizes that a docx file that was not considered to be informative in an early stage may be very informative during later stages of acquisition and can contribute to the detection model's performance and the updatability of the anti-virus as well.

We showed that during very early stages, both of our AL methods acquired 304 malicious docx files out of 312 present in the pool (97.4% of the malicious docx files) and both updated the detection model and anti-virus signature repository. This was achieved on the 12$^{\text{th}}$ trial, while it took the Random method 270 trials to acquire the same number of malicious docx files, and the SVM-Margin required the entire pool, or 280 trials. Ultimately, ALDOCX provided a reduction in the labeling efforts of 95.5% and 95.7%, respectfully with Random and the SVM-Margin, and updated both the detection model and the anti-virus software with new informative docx files—benign and especially malicious which are more valuable for detection purposes. The results of this third and last experiment prove that our framework can be also deployed inside organizations for creating a detection model that is well tailored toward the proclivities of the organization itself—according to the docx files it uses and creates. It would be advantageous for every organization to have its own set of benign files on which the detection model is trained upon—by doing this, the detection rates would increase, and false alarms rates would decrease. We also compared the detection rate of our methods with the best 10 anti-viruses commonly used by organizations. The best detection rate, 85.9%, was provided by *TrendMicro*, while our methods outperformed all the anti-viruses in the task of detecting new malicious docx files. Using the SVM classifier with 100 structural features (SFEM), we achieved 93.4% which required using 90% of the dataset for training (due to the 10XV settings). Using the full ALDOCX framework including the Active Learning methods and its enhancement process, we even improved the performance of SVM-SFEM, achieving a 94.4% TPR using only 14% of labeled data (2,011 docx files out of 14,000), which means a reduction of 84.4% in labeling efforts. In future work, we are interested in extending this framework to the detection of additional Office files such as xlsx, pptx which share the same XML based structure as the docx file. We also interested in integrating ensemble learning [18] techniques to enhance the detection capabilities, and document instrumentation techniques [19] that have been used to successfully detect malicious PDF files.

References

[1] T. Schreck, S. Berger, and J. Göbel, "BISSAM: Automatic vulnerability identification of office documents," in *Proc. 9th Int. Detection Intrusions Malware, Vulnerability Assessment Anonymous*, 2013, pp. 204–213.

[2] J. Dechaux, E. Filiol, and J. Fizaine. (2010). *Office Documents: New Weapons of Cyberwarfare*. [Online]. Available: http://2012.hack.lu/archive/2010/Filiol-Office-Documents-New-Weapons-of-Cyberwarfare-paper.pdf

[3] H. K. Jnanamurthy and C. W. S. Singh, "Threat analysis and malicious user detection in reputation systems using mean bisector analysis and cosine similarity (MBACS)," in *Proc. Annu. IEEE India Conf. (INDICON)*, Dec. 2013, pp. 1–6.

[4] N. Nissim, R. Moskovitch, L. Rokach, and Y. Elovici, "Detecting unknown computer worm activity via support vector machines and active learning," *Pattern Anal. Appl.*, vol. 15, no. 4, pp. 459–475, 2013.

[5] N. Nissim, R. Moskovitch, L. Rokach, and Y. Elovici, "Novel active learning methods for enhanced PC malware detection in windows OS," *Expert Syst. Appl.*, vol. 41, no. 13, pp. 5843–5857, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2014.02.053.

[6] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, p. 27, 2011.

[7] N. Nissim *et al.*, "ALPD: Active learning framework for enhancing the detection of malicious pdf files aimed at organizations," in *Proc. JISIC*, 2014, pp. 91–98.

[8] N. Nissim, A. Cohen, C. Glezer, and Y. Elovici, "Detection of malicious PDF files and directions for enhancements: A state-of-the art survey," *Comput. Secur.*, vol. 49, pp. 246–266, Nov. 2014.

[9] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, Nov. 2001.

[10] R. Moskovitch, N. Nissim, and Y. Elovici, "Malicious code detection using active learning," in *Privacy, Security, and Trust in KDD*. Berlin, Germany: Springer, 2009, pp. 74–91.

[11] R. Herbrich, T. Graepel, and C. Campbell, "Bayes point machines," *J. Mach. Learn. Res.*, vol. 1, pp. 245–279, Aug. 2001.

[12] Y. Baram, R. El-Yaniv, and K. Luz, "Online choice of active learning algorithms," *J. Mach. Learn. Res.*, vol. 5, pp. 255–291, Mar. 2004.

[13] H. Kiem, N. T. Thuy, and T. M. N. Quang, "A machine learning approach to anti-virus system," in *Proc. Joint Workshop Vietnamese Soc. AI, SIGKBS-JSAI, ICS-IPSJ IEICE-SIGAI Active Mining*, Hanoi, Vietnam, 2004, pp. 61–65.

[14] N. Srndic and P. Laskov, "Detection of malicious PDF files based on hierarchical document structure," in *Proc. 20th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2013, pp. 1–16.

[15] N. Nissim, R. Moskovitch, O. Barad, L. Rokach, and Y. Elovici, "ALDROID: Efficient update of Android anti-virus software using designated active learning methods," *Knowl. Inf. Syst.*, vol. 49, no. 3, pp. 795–833, 2016.

[16] N. Nissim, A. Cohen, and Y. Elovici, "Boosting the detection of malicious documents using designated active learning methods," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Miami, FL, USA, Dec. 2015, pp. 760–765.

[17] R. Amin, J. Ryan, and J. Van Dorp, "Detecting targeted malicious email," *IEEE Security Privacy*, vol. 10, no. 3, pp. 64–71, Nov. 2012.

[18] H. V. Nath and B. M. Mehtre, "Ensemble learning for detection of malicious content embedded in PDF documents," in *Proc. IEEE Int. Conf. Signal Process., Informat., Commun. Energy Syst. (SPICES)*, Kozhikode, India, Jun. 2015, pp. 1–5.

[19] D. Liu, H. Wang, and A. Stavrou, "Detecting malicious javascript in PDF through document instrumentation," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Atlanta, GA, USA, 2014, pp. 100–111.

[20] R. Moskovitch, N. Nissim, and Y. Elovici, "Malicious code detection and acquisition using active learning," in *Proc. IEEE Intell. Secur. Informat.*, May 2007, p. 371.

[21] R. Moskovitch, N. Nissim, and Y. Elovici, "Acquisition of malicious code using active learning," in *Proc. 2nd Int. Workshop Privacy, Secur. Trust (KDD)*, 2008, pp. 1–9.

Authors' photographs and biographies not available at the time of publication.