Detection of Malicious OOXML Documents Using Domain Specific Features

Thesis · June 2019

DOI: 10.13140/RG.2.2.13009.40801

1 author:

Priyansh Singh
Indian Institute of Technology Delhi
2 PUBLICATIONS **52** CITATIONS

SEE PROFILE

Detection of Malicious OOXML Documents Using Domain Specific Features

by

Priyansh Singh

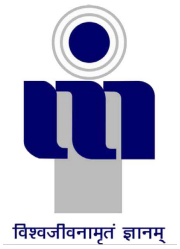
2017IS - 17

A thesis submitted in partial fulfilment of the requirements for the award of the degree of

Master of Technology in

Information Security

2017-19



ATAL BIHARI VAJPAYEE-

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT

GWALIOR - 474015, MADHYA PRADESH, INDIA

Certificate

I hereby declare that the work, which is being presented in the dissertation, entitled Detection of Malicious OOXML Documents Using Domain Specific Features, in fulfilment of the requirement for the award of the degree of **Master of Technology** in **Information Security** and submitted to the institution is an authentic record of my own work carried out during the period *July-2018* to *May-2019* under the supervision of Prof. Shashikala Tapaswi. I also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

_____

Mr Priyansh Singh

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

_____

Prof. Shashikala Tapaswi

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Candidate's Declaration

I hereby certify that I have properly checked and verified all the items as prescribed in the check-list and ensure that my thesis is in the prop er format as specified in the guideline for thesis preparation.
I declare that the work containing in this rep ort is my own work. I understand that plagiarism is defined as any one or combination of the following:

(1) To steal and pass off (the ideas or words of another) as one's own

(2) To use (another's production) without crediting the source

(3) To commit literary theft

(4) To present as new and original idea or product derived from an existing source.

I understand that plagiarism involves an intentional act by the plagiarist of using someone else's work/ideas completely/partially and claiming authorship/originality of the work/ideas. Verbatim copy as well as close resemblance to some else's work constitute plagiarism.
I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programmes, experiments, results, websites, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.
I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report/dissertation/thesis are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable. My faculty supervisor(s) will not be responsible for the same.

Signature:

Name: Priyansh Singh

Roll. No: 2017IS-17

Date:

Abstract

Since the beginning of software development, these have been a counter-culture of breaking the mould and using the software for nefarious purpose or personal gain. Through subliminal understanding over the years, the average user understands, portable executables from unidentified sources can be malicious, but the same knowledge has not been transcended for documents.

Various antivirus software vendors, as well as independent researchers, have noticed the increasing trend of malicious document campaigns. Year after year, there have been reports of a steady increase in these campaigns, including payloads that have malicious macros, embedded executables, sniffers and even ransomware attached in them. Traditional signature-based detection fails in light of the unprecedented level of sophistication in these attacks. Moreover, since these attacks use documents, it is hard for traditional signature-based detection to identify malicious code embedded with the user content.

In this thesis, a new method for automated detection of malicious office documents is proposed that puts domain knowledge of the Office documents first. It introduces the use of forensic identifiers and metadata for the detection of malicious documents. These features alone lead to high accuracy results while when combined with format agnostic structural features provide more robust representation and similar results. When compared to the previous state-of-the-art methods in this domain, the proposed work either over-performed or performed at par with these methods. Additionally, the use of forensic identifiers opens the dimension of customising the implementation as per the deployment environment, which was not possible until now.

Acknowledgments

I am highly indebted to **Prof. Shashikala Tapaswi** and obliged for giving me the autonomy of functioning and experimenting with ideas. I want to take this opportunity to express my profound gratitude to him not only for his academic guidance but also for his interest in my report and constant support coupled with confidence boosting and motivating sessions which proved very fruitful and were instrumental in infusing self-assurance and trust within me. The nurturing and blossoming of the present work is mainly due to his valuable guidance, suggestions, astute judgement, constructive criticism and an eye for perfection. She always answered a myriad of my doubts with smiling graciousness and immense patience, never letting me feel that I am novices by always lending an ear to my views, appreciating and improving them and by giving me a free hand in my report. It's only because of her overwhelming interest and helpful attitude, the present work has attained the stage it has.

A special plea of gratitude to Mr Sanchit Gupta at SAG, DRDO, Delhi for introducing me to the topic of reverse engineering and malware analysis. Thanks to my family and friends, Mr Aditya Sharma, Ms Dharvi Verma, Mr Divyanshu Sharma, and Ms Hanniyah Mir, for their constant support interesting discussions, shared frustration and laughter.

Finally, I am grateful to our Institution and colleagues Dr Neha Agarwal, Ms Anjula Mehto, Mr C.P. Sharma, Mr Debabrata Sharma, Mr Mandar Patil, and Ms Prateeti Jain whose constant encouragement served to renew my spirit, refocus my attention and energy and helped me in carrying out this work.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

This chapter presents a broad overview of the thesis and provides context on why research into the detection of malicious office documents not only crucial but also necessary at the same time. Section 1.1 provides a reference malware detection domain. Section 1.2 projects the malicious actor problem in the light of malicious office documents. Next, in Section 1.3 the objectives of the thesis are presented, while in Section 1.4 the research flow with followed is introduced. Section 1.5 outlines the rest of the thesis.

1.1 Context

With the rapid infusion of technology in everyday lives, humanity has redefined how work is accomplished. Most if not all industries rely on different software systems to function. The use of mobile phones and the internet has become ubiquitous to people in every walk of life. This inadvertent dependence on technology has also led to absolute trust as well. Users generally agree to outrageous terms of service agreements and are willing to share personal information. This information is a valuable asset for whole industries like marketing agencies, recommendation systems and malicious actors. The numerous services the users readily trust, sell their information to other interested entities. Even if the software and services used are respectful of the user's data, there is always the probability of being on the wrong end of vulnerability or exploit. Cybercriminals and attackers alike, use various tool-kits and vulnerabilities available to gain access to vulnerable systems. Such activities require minimal effort from the criminals and are largely profitable for

criminals. The ransomware attacks of Locky and WannaCry have led to the loss of billions of dollars across the globe. As a consequence of this, the cybercriminals push the envelope for evolution and variation of malicious software and other attacks propagated over the web.

With the increase in users available over the internet, the attack surface of potential attacks has also rapidly increased. Most users on the internet are either ignorant or unaware of cybercrime and how it is perpetuated. With experience, novice users become wary of executables downloaded from unrecognised sources, as they may contain malware, but they do not hold the same depth of mistrust for other file types. This bias is one of the primary reasons why there has been a resurgence of malicious documents in the last decade. Malicious campaigns use social engineering techniques to motivate users into executing malicious code.

Additionally, most detector methods and antivirus software still largely depend on signature-based and heuristic based methods. Signature-based methods maintain a dictionary of signatures updated frequently by a software vendor with new signatures. As mentioned in the book Gray Hat Hacking [4], the onslaught of new malware released was so elevated in 2009, Symantec was required to write new virus signatures every eight seconds. Malicious campaigns also employ obfuscation techniques where the malware can encrypt parts of itself, thereby changing its signature. Heuristic-based methods [5] on the other hand uses features like API calls, OpCode, and Control Flow Graphs They can detect the family of malware with ease using generic signatures and previously indicated features. Both of these methods fail when challenged by novel malware or zero-day vulnerabilities.

Automation makes the generation of malicious programs relatively rudimentary. Malware authors have access to state-of-the-art tools like Metasploit and many malware generation toolkits with which they can not only automate the generation of malware but also include evasion countermeasures like, encryption, obfuscation, behavioural spoofing, and configuration based countermeasures. Launching a full campaign has also been primarily streamlined. Phishing emails and spam prays upon the gullibility and the lack of awareness of the user as mentioned in [6]. Even though there has been a decline in spam email messages, the rate of malicious spam has increased to 26% [7]. Moreover, Verison's 2018 Data Breach Investigations Report [8] reported that 92.4% of

malware is delivered to the target through email.

Most document file formats are designed to provide a similar experience across hardware configurations with platform independence. Portable Digital Format (PDF) files, Office documents, Rich Text Format (RTF) documents, Flash documents would be some examples. These file formats provide flexibility in representation and features like embedded content like other documents, images, and videos, and active scripting like JavaScript, Action Script, and Visual Basic for Applications (VBA). These features make documents ripe for use in malicious campaigns. Most of the implementations allow the script writer to subvert user knowledge by executing scripts in the background. Most of these are 'features' and not vulnerabilities, the organisation is unable to eradicate them, and at best can limit their usage, just like Microsoft (MS) did by asking users for permission on any document that has a macro associated with it. However, these measures are limited and are often vitiated by malware authors.

In the SophosLabs 2019 Threat Report [9], the authors have noted an increase in targeted attacks. Targeted attacks are unpredictable and blend in easily with other incoming traffic. In such cases, attackers first scout the target for access or open vulnerabilities. The security community releases several Proofs-of-Concept and open vulnerabilities regularly in everyday applications such as Adobe Acrobat Reader (CVE-2018-4895, CVE-2017-16398, CVE-2018-4917), Foxit Reader (CVE-2017-10994), and the whole of Microsoft Office Suite (CVE-2018-8162, CVE-2018-1030, CVE-2017-8550, CVE-2018-8157). An attacker uses social engineering and phishing emails, to bait the users into executing malicious code. It has also been noted, the new vulnerabilities and proof of concepts are simpler to replicate thus making it undemanding for attackers to replicate and launch campaigns.

1.2 Problem Domain

The Microsoft Office applications such as Word, Excel, Powerpoint, and Outlook received critical praise since their launch. These applications became commonplace in offices and personal use cases. With a large install base and heavy use for communicating, it became a hotbed for

criminal activities. One of the early examples of MS office getting/being adopted for malice and mass hysteria is, Melissa. In 1999, bad actors created a malicious macro infection which attached to MS Word documents on a user's machine [10], which analyst still consider one of the most massively distributed malware adds itself to the user's macro definition library and itself to up to 50 people in their contact list. It led to mass hysteria and loss of thousands of work hours. It is a widely accepted fact that despite security regulations placed by Microsoft, an attacker can bypass them. [11] showed, for an older version of Microsoft Office and Open Office, techniques through which an attacker can bypass default security measures preventing macro execution.

Unlike malicious documents that were created in the 1990s and early 2000s, attacks that leverage office documents today are often obfuscated or encrypted and thus harder to detect. Take for example the attack on Western Ukraine's power supply in 2016 [12, 13]. The attackers exploited a known vulnerability from 2014 and malicious macro to bring down three power distribution centres and nearly 60 substations and to leave more than 230,000 residents without power. McAfee Labs Threat Report 2018 [14], total macro malware has seen a steady increase since Q2 of 2016. While introducing a more granular security control for the macro in their enterprise vertical in 2016, Microsoft noted, 98% of Office-targeted threats use macros [15]. Since then the trend has shifted towards new attacks and vulnerabilities in subsystems like the equation editor.

In the early, to mid-2010s there was a resurgence of malware that used PDF files to launch attacks. This trend has mostly shifted from other document formats to various office document formats as reported by Cisco [16]. Most modern document formats provide malicious authors with a playground full of features and vulnerabilities to exploit. At the Security Analyst Summit in 2019, researchers from Kaspersky Labs reported an increase in attacks using Office documents. In a presentation they reported as of Q4 2018, 70% of attacks detected by their software now target Office vulnerabilities [17]. These numbers have increased from 16% in Q4 2016. As this was not alarming enough, the malware authors target legacy components present in the Office suite to target users. The article goes on to report CVE-2017-11882 and CVE-2018-0802, vulnerabilities present in the equation editor module of the suite, are two of the most exploited vulnerabilities. Recorded

Future in reported [18], Microsoft was targeted by eight out of ten most exploited vulnerabilities, and of these eight vulnerabilities, six of them used components of the Office suite to initiate the assault.

Aside from the more traditional attacks, malicious documents are being used to drop ransomware payload. SophosLabs 2018 Malware Forecast [19] mentions the use of MS Word files to spread ransomware of the Locky family. In a McAfee Report from 2018 [20], the authors indicate the use of malicious documents by the cybercrime group Lazarus for reconnaissance of users mining cryptocurrency. The creation of malware is further facilitated by the many exploit kits and automation methods; some of these are discussed in Section 2.1.2.

1.3 Objectives

The objective of the research is to analyse the current state-of-the-art methods in detection of malicious office documents along with various antivirus systems and develop a new methodology which is scores high on accuracy and other metrics and at the same time is adverse to novel attacks. This work proposes a model which can be customised as per the need and the domain knowledge of the scenario. Consequently, during the thesis, there are several objectives need to be achieved:

Detection of malicious documents: To develop and implement features and feature extraction strategy based on domain knowledge of Office document formats. These features are paired with numerous machine learning models to identify and categorise documents in malicious and benign classes seamlessly and efficiently.

Evaluation Against Novel Attacks In a realistic scenario, not all the possible malware configurations are known. Furthermore, new vulnerabilities, malware mutate, and novel malware campaigns are discovered every week. The model developed needs to be tested against attacks which are unknown to the system at the time of training.

1.4 Research Workflow

Based on the research objectives, the following workflow was followed to produce results:

Step 1 Documents both malicious and benign are collected from trusted sources such as VirusTotal [21] and CommonCrawl [22]. The benign files were tested against antivirus software to confirm their integrity. The collected files were analysed for identifiers and domain knowledge was used to recognise features.

Step 2 An static extraction method was developed to extract the identified features without execution of the file. These features were studied and indexed. The hypotheses were created around these features for evaluation. The methodology is described in detail in Chapter 3.

Step 3 Machine learning models were selected in order to convert the knowledge extracted into tangible decisions. Linear SVM, RBF SVM, Random Forest and XGB Classifier were tried for classification of the malware.

Step 4 To test against novel attacks, the document set is split into sections around the *timeSinceCreation* feature. The feature represents a numeric value of time in seconds since the file was created to the date the file was extracted. Files that were created before January 2018 were put in the training set while the rest were considered as testing data.

1.5 Thesis outline

The thesis is divided into four other chapters. A summary of the chapters is presented below.

- Chapter 2 presents related research in the domain of malicious office documents; both published literature and available tools. The chapter also summarises the characteristics and structure of Office document formats.

- Chapter 3 explains the proposed hypothesis; it also covers the architecture, features selected, extraction methodology, feature selection methodology and machine learning algorithms.

- Chapter 4 to evaluate the proposed objectives, three experiments are proposed. Chapter 4 describes these experiments in detail, along with values for hyper-parameters. Results from each section are also discussed in detail.

- Chapter 5 concludes the thesis. It highlights the contributions by the author, limitations of the methodology and the future scope of the domain.

Chapter 2

Literature review

In this chapter, the body of literature so far is summarised and discussed. This includes the MS Office formats, static and dynamic analysis techniques, malicious tool-kits for the creation and the research produced for increasing the detection rate of the Office malware. The key research is discussed in detail, analysed, and research gaps are identified. The problem tackled throughout this thesis is formulated, and a brief conclusion summarises the chapter.

2.1 Background

As discussed earlier in Sections 1.1 and 1.2, the trend of attacks has moved from executables and towards the use of more trusted sources like documents. This section also discusses some popular method of attacks on unsuspecting users. Toolkits for both creation and analysis of malware are also discussed. File formats which facilitate the creation of are also put under the microscope.

2.1.1 Office Malware

The Office malware uses features and scripting languages that are part of the software stack. These various point-of-entries provide an outstanding amount of flexibility for malicious campaigns and make it tremendously difficult for detectors to identify correctly. In modern Office applications, the malware can be propagated through any of the following three methods.

(1) **Features:** In the software industry, features are how one vendor differentiates from the other, and to remain competitive they update their software with new features. Most often,

this leads to a list of deprecated features which are no longer under software development and new features which have recently been introduced. Malicious actors sequester these features and use them to propagate malice. Since the legacy software is no longer in development but has supported through backwards compatibility in newer releases of software, they make an ideal target to launch an attack. Take, for example, Dynamic Data Exchange (DDE) links, which were introduced in early the 1990s to facilitate interprocess communication. The modern office document stack still supports these links even though they have been replaced by Object Linking and Embedding (OLE). Malicious actors have used DDE attacks to evade detection and execute the malware on the victim's device. FIN7 used this technique to attack the banking sector using macros and DDE links [23]. Newer features like access to PowerShell from a document has been indiscriminately abused by malware authors as well. With this category of attacks, the software vendor cannot mitigate these attacks themselves and depend on external resources like anti-virus software to detect and remove malware.

(2) **Macro:** The Microsoft Office suite supports the use of macros, which are small scripts that provide automation functionality. These macros can either be recorded, linked to external resources or scripted in Visual Basic for Applications (VBA). This feature has taken a life of its own, and it was one of the most prominent methods of attack in the mid-2010s. Malware authors could write their script of both downloader and dropper kind and execute on the victim's computer at the time when the document is opened. In the CFB formats, which are still in use today, there were no controls over the execution of VBA script. Whereas with the newer OOXML file formats, if the document contains macro or other embedded content the software asks for the user's permission before execution. Microsoft has taken other initiatives as well to curtail the impact of macro-malware. They introduced more granular controls for enterprise Office 2016 which allowed administrators "to selectively scope macro use to a set of trusted workflows" and provide users with stricter

and different warnings when not in an approved workflow scenario [15].

Emotet is a popular banking trojan which has started using malicious macros. The recent emotet campaigns use social engineering, sending malspam to users with billing information and past due notices, to execute the macro. Emotet then modifies registry keys and attaches to other processes. The Trojan contacts the C2 server to register a new infection and receives further instructions.

(3) **Vulnerabilities:**

The point has been made time and again that software vulnerabilities are the key reason why software fails under attacks. Each vendor updates their software to fend off attackers, but most times endpoint fails to update and is thus left vulnerable.

Take CVE-2017-11882 and CVE-2018-0802, for instance, because of improper handling of objects in memory, an attacker can run arbitrary code on the target machine, and if the user has sufficient privileges, they could even take control of the machine remotely.

There are numerous tools which help support the actions of malicious actors; they provide the ability to create advanced malware and automate the production of malware of a type. These toolkits are discussed in Section 2.1.2. There are numerous other tools also available which support malware analysts in identifying malware and making decisions. These toolkits are described in Section 2.1.3.

2.1.2 Existing Exploit Toolkits for Creation of Malicious Documents

The proliferation of office malware has become rampant and tools, and utilities for creation of these files are much more prevalent. Much of the blame for this rapid dissemination to the masses belong to the numerous exploitation toolkits. The tools are developed either to automate the manual, tedious tasks or to extend the knowledge to the masses. In the case of the latter, the tool kits are made commercial and sold to other malicious actors.

As stated in [24], Microsoft Word Intruder (MWI) is one such infamous and utility which in the past few years has gained traction. It gives the attacker the ability to stack the use of few vulnerabilities on top of each as well as to create decoy documents. The proliferation of the malware is achieved via a dropper, i.e. the malicious document is embedded with a portable executable, or a downloader, i.e. the payload executes and downloads from the embedded URL and executes the file. [25] reported the presence of command and control capabilities in the toolkit.

The white paper on Office Exploit Generators by Gabor Szappanos [26] enumerates a few other exploit kits floating in the marketplace. AKBuilder and its two variants AK-1 and AK-2 have been quite popular in the past. As the report states, AK-1 allows the attacker to create downloader and dropper files with options for decoy features. A downloader malware runs scripts on the victim computer using features like PowerShell to download malware at runtime, while a dropper malware is contains packed malware with the file at the time of execution. AK-1 employed many exploits at the same time namely CVE-2012-0158 and CVE-2014-1761 and system APIs such as ShellExecuteA & WinExec. While the AK-2 uses a single vulnerability, CVE-2015-1641, and WinExec for its execution, it has the same decoy capabilities but no longer supports downloaders. Both AK-1 and AK-2 follow stringent hierarchy while forming malicious documents. This hierarchy and their exact method of triggering them have been studied in [26].

In the white paper [27], the Sophos team analyses the functioning of seven Office exploit generators and their functioning against one another. Furthermore, amateur attackers do not even need to surf the dark web for malware. Tools like Metasploit and numerous projects like Malicious Macro Generator and Generate-Macro are available to them for one of the attacks. With time there has been a trend towards discovering vulnerabilities which are easy to replicate from Proofs-of-Concept (PoCs), this gives ammunition to the malicious actors while the software vendors rarely have time to respond [28].

2.1.3 Existing Toolkits for Analysis of Malicious Documents

There are numerous tools available for a malware analyst to automate the detection of malware. These tools rely on some signature-based, behaviour-based and heuristic-based methods such as those mentioned in [5] and [29]. While performing the analysis manually, the analyst needs to follow some key steps laid out by the Lenny Zelster [30] and SANS Institute.

(1) Examine the document for irregularities

(2) Find the location and extract of embedded suspicious code or objects

(3) Depending on the nature of the suspicious code,

(4) If it is a scripting language like JavaScript or VBA, deobfuscate and examine

(5) Else, it is shellcode or portable executable then disassemble and debug

The analysis techniques are classified into static analysis and dynamic analysis [31]. An analyst employs both of these methods while trying to discern whether a document is malicious or not along with the intentions of the document. While performing static analysis, the analyst never executes the file; instead, they extract strings, suspicious code, and hex-dump. Whereas during a dynamic analysis based on the information gathered from the static analysis, sections of the malware are executed in a sandboxed environment.

2.1.3.1 Static Analysis

Depending on the file under examination, there are a whole deck of tools available which can assist in a static analysis. The MS Office document formats can be subdivided into Compound File Binary Format (CFBF) and Office Open XML (OOXML) formats. Sections 2.1.5 and 2.1.4 discusses the nuances of the file formats in detail. During static analysis, the analyst examines the potentially malicious file without ever executing it. They focus on the structure, metadata and the embedded content stored in the document. The goal is to determine if a file is malicious without

ever executing the code embedded in it. Over the years, there have been several tools which help with the analysis process.

OfficeMalScanner, is a utility which can be used for static analysis of malicious documents of either format. It is limited in its implementation and has not been updated in nearly a decade. It can be used to expand and extract parts of the OOXML file. It also supports extraction of shellcode and VBA scripts. OfficeDissector was one of the first parser explicitly designed for security analysis of OOXML documents. It can be used to extract all embedded contents present in an Office Document. There was a recent release of Vba2Graph utility that generates a VBA call graph based on static analysis. Figure 2.1 provides a graphical representation of a sample macro designed to execute only on a machine running windows. This was adapted from the graphs produced using the Vba2Graph utility.

oletools [32] library was written in python are actively under development. The library supports CFBF with partial support to OOXML. It has numerous tools like oleid, olevba and oleobj, which can be used to identify, extract, and analyse files embedded by Object Linking and Embedding (OLE). Based on patterns and heuristics, it also recomends the malice of the document. The tool library also includes support for newer attacks based on Dynamic Data Exchange (DDE) using the msodde tool. olevba tool [32] of the library can be used to extract and analyse VBA macros. It has subprocesses like MalHost-setup, which allows for the extraction of shellcode and embeds it in a portable executable (PE) for further analysis.

2.1.3.2 Dynamic Analysis

There are only a small number of tools that allow for dynamic analysis of Microsoft Office documents. Most sandbox environments allow the execution of Office files in a limited capacity. Cuckoo sandbox, anlyz.io, VirusTotal, hybrid-analysis, and Cryptam are some online dynamic analysis tools. The analyst submits their file to one of these services, depending on the service analysis is performed. VirusTotal tests the files against numerous antivirus services and results in a detection report. any.run executes the file in a sandbox and lets the analyst look through the

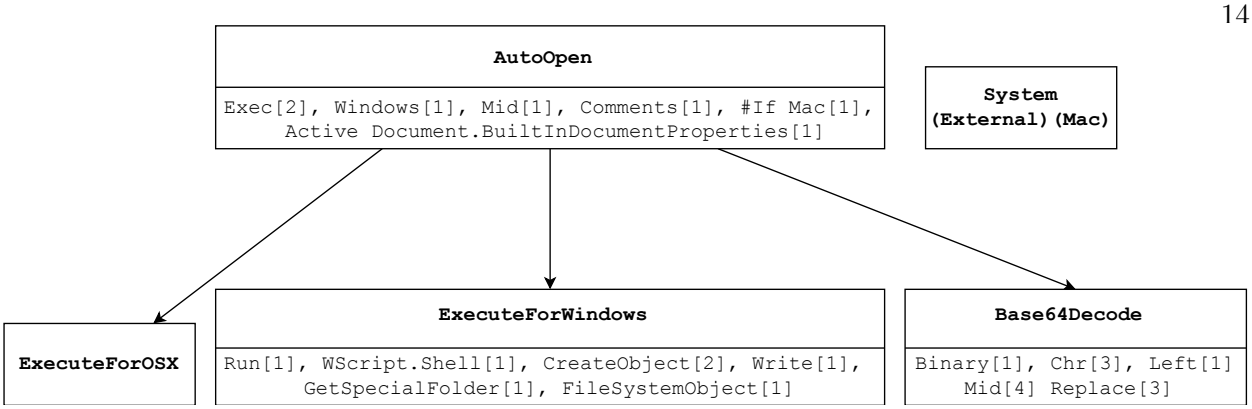


Figure 2.1: Graphical representation of a sample Macro created using VBA

process and network traffic that preceded.

Macros have been in the past the key for various attacks, and such there are numerous dynamic techniques to detect and analyse them. Lazy Office Analyser tries to neutralise any obfuscation present in the document. It extracts and analyses VBA scripts, JavaScripts and URLs from the document. ViperMonkey is an open source tool which can be used for emulation of VBA files in a sandboxed environment. It has the capability to de-obfuscate VBA macros. vhook is another detector which looks for suspicious indicators like HTTP.Open, URLDownloadToFileA and other API calls.

2.1.4 Compound File Binary Format (CFBF) Schema

Every malware has some representation in the file structure and schema as well. Thus it is necessary to understand the schema and build domain knowledge of the format before attacking the problem. The CFBF has been a accessible file format in the past and was actively used by suites 97-2003. They were sometimes also referred to as OLE Compound Document Format [33]. The structure of the format mirror that of the File Allocation Table (FAT) file system where blocks are defined and assigned to many allocation tables

The format was designed keeping in mind future requirements as such it allowed for the use of various independent data streams. These data streams provided the capability to store and execute compiled programs (executables), media files, such as images, videos and animations, and

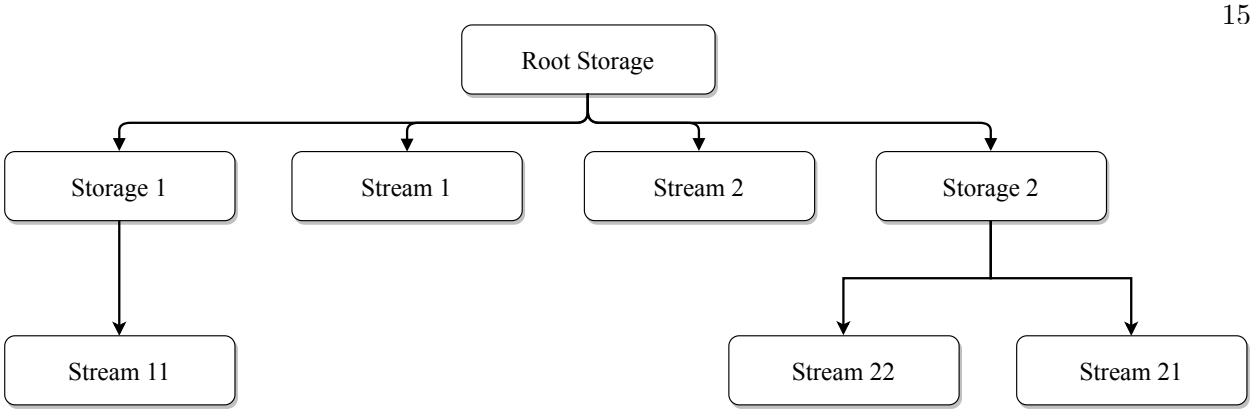


Figure 2.2: Example of hierarchy followed by CFBF documents adapted from [3]

compressed files amongst the others. The compound binary file format was used widely across Microsoft products. Office file formats, installers and windows thumbnails follow it.

[3] describes a CFBF document as, a hierarchical structure like a file system for data streams. There is a Root Storage which may link to other storage blocks or content streams. Each content stream gets divided into sectors, and these sectors are sometimes stored out of order as use sector chains to maintain coherence. Figure 2.2 provides a graphical representation of this hierarchical structure. The CFBF header is of 512 bytes precisely and located in the first sector of the document. It is often used to identify the type of file under inspection, along with markers such as a unique identifier, revision number size of a sector, the total number of sectors and part of the master allocation table and sector allocation table. These tables record identification information of where the data is stored in the file schema and provide access to the reader when required. As part of the metadata timestamps for created, last modified and last printed are also stored in the document. The schema itself does not provide with any special measures against malcode bearing documents, nor does it any steps to mitigate the damage after execution. The onus of both lies on the application and the system administrator.

2.1.5 The Office Open XML (OOXML) File Format

The Office Open eXtensible Markup Language (XML) format was released in 2006 and has mostly replaced the older compound file binary format. With the launch of Office 2007, the OOXML format also became an ECMA standard (ECMA-376) and since has been revised four times, most recently in 2016. The OOXML files follow a ZIP standard, where all the files are stored in a ZIP file with the relevant extension [1]. Each of the files or package contains XML files as well as may contain some UTF-8 or UTF-16 encoded parts. It supports Multipurpose Internet Mail Extensions (MIME) standard and stores the files along with the schema. Images, scripts and other media files stored in the document are also stored inside the same package. The [content_type] file provides MIME type metadata for parts of the package. '_rels' contains all the relationships between various objects in the document in the package. 'docProps' folder contains properties related to the OOXML document and 'word' folder contains the core part of any Word document [34]. This subsection discusses the file structure and the metadata of the OOXML document in detail.

2.1.5.1 The OOXML File Structure

When an OOXML package is unpacked, it must have the *[Content_Types].xml* as well as the relationship schema files. The *[Content_Types].xml* file is found at the root level of the package and specifies every part of the document along with its name and type. Along with *[Content_Types].xml* each OOXML package contains a *'_rels'* folder at root level. This folder and its contents represent all the relationships a document maintains within and outside of the package. The *'.rels'* file is used by the file structure to specify the type, the target file, target mode and assign a relationship id from which the target file belongs. Depending upon the document and its contents several sub *'_rels'* folders and *'.rels'* files maybe created. For instance, if there are images or separate header or footer in the word document, the *word/document.xml* file will spawn its own relationships folder and files.

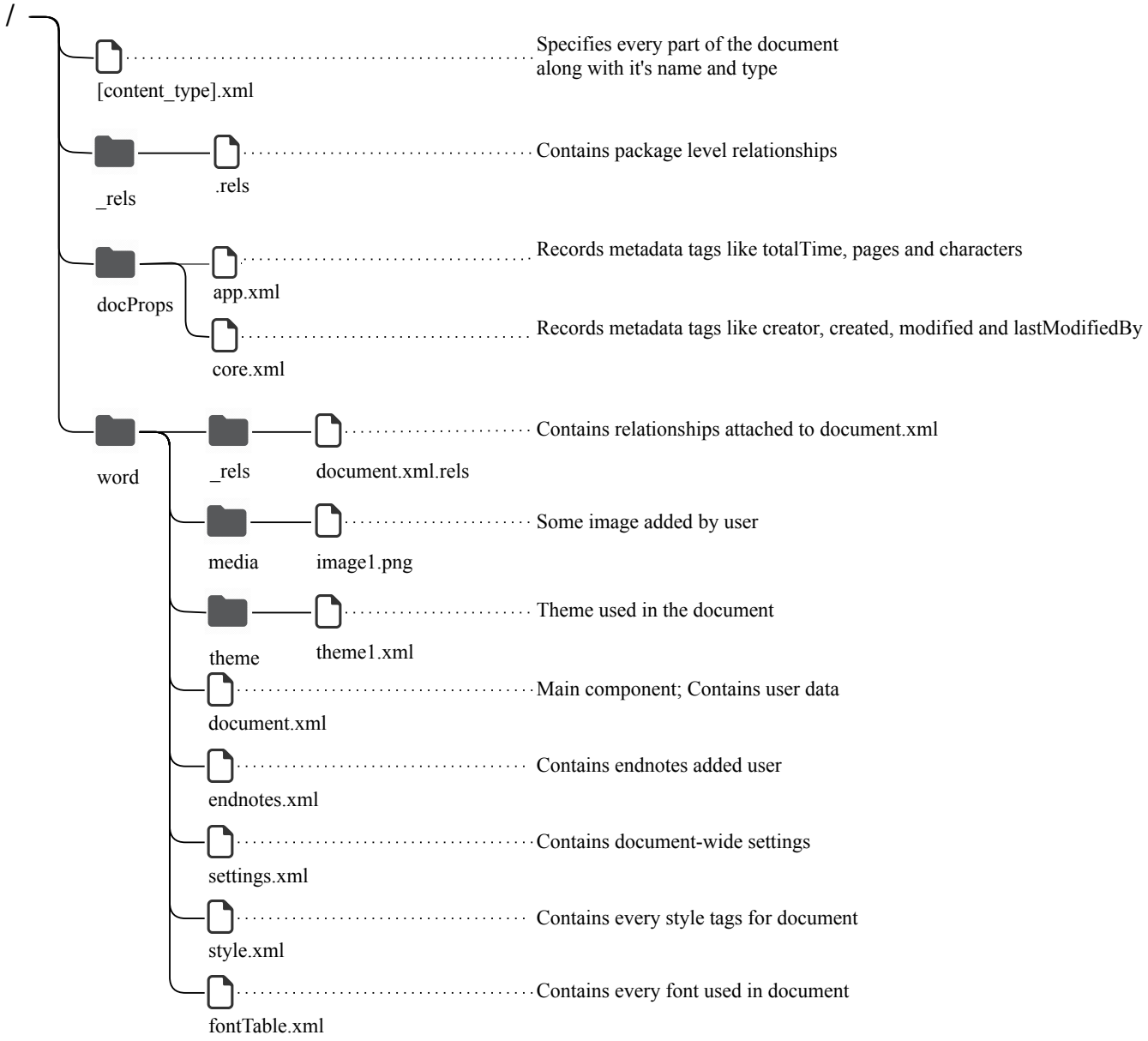


Figure 2.3: Structure of a sample OOXML file

Each of these packages also contains *docProps* folder containing *docProps/app.xml* and *docProps/ core.xml* files. These documents contain application and document specific properties and are further discussed in Section 2.1.5.2. With the case of a WordProecessingML document the root also contains *'word'* folder. This folder, depending on the content of the document, may contain numerous XML files and binaries. *word/document.xml* file consists of the body of the document. Header,

footer, endnotes, glossaries, settings or web-settings amongst others reside in their separate XML file. If there are multiple headers, footers and other components in a document, they are enumerated and stored separately. Macro and other embedded content are also stored in this same folder. All of the files in */word/* folder are also represented in *word/_rels/document.xml.rels*. Figure 2.3 shows the file structure, along with a brief description of each sub-file.

2.1.5.2 Metadata stored in OOXML Documents

Unlike executables where the metadata refers to the data around the container of executable, part of the metadata in OOXML documents is stored within the documents in the form of two core XML files namely *core.xml* and *app.xml*. The *app.xml* file records details like name of the application and application version, level of security enforced by the document, number of lines, characters, paragraphs present in the document. Table 2.2 enumerates all possible values recorded by *app.xml*.

Table 2.1: Metadata recorded by core.xml, adapted from ECMA Standard-376 [1] and Didriksen [2]

Element	Description
category	Category of the document, like "important"
contentStatus	Status of the document, like "final"
created	Timestamp of creation of document
creator	User Identity
description	Description of content
identifier	Identification reference to some resource eg URL
keywords	Keywords for searching
language	Language information of document
lastModifiedBy	Identity of user who last modified
lastPrinted	Timestamp of last print
modified	Timestamp of last modification
revision	Number of revisions on document
subject	Topic of document
title	Name of the document
version	Version number of document

The *core.xml* file records timestamps of when the document was created, last modified and

last printed. It also stores the information of the creator of the document, number of time it was modified title, category language amongst other things. Table 2.1 describes all these elements and their respective significance. If there is any custom metadata, it is stored in the same folder under the title of *custom.xml*. This file provides vital information which is not represented anywhere else in the schema. It is the role of the application used to create the document to record these details.

In their thesis [2], Didriksen, compared some publicly available tools such as EnCase Forensic and Forensic Toolkit (FTK) and noted their severe limitations. Out of possible 49 tags, EnCase Forensic could only extract 14 of the tags while FTK could extract 20 tags, other tools like DSO Tool are even more limited and extract only 5 of the tags.

Table 2.2: Metadata recorded by app.xml, adapted from ECMA Standard-376 [1] and Didriksen [2]

Element	Description
application	Application used for creation of document
appVersion	Version of application used
characters	Total number of characters in the document
charactersWithSpaces	Number of characters including spaces
docSecurity	If the document is password protected or not
digSig	If the document has been signed or not
hLinks	Lists all hyperlinks in the document
hyperlinkBase	Base URI string for hyperlinks
hyperlinksChanged	List of hyperlinks changed in the last save
lines	Number of lines in the document
linksUpToDate	If hyperlinks in the document are updated
mmclips	Number of multimedia clips
pages	Number of pages
paragraphs	Number of paragraphs
properties	Application-specific properties of file
template	If some template was used to create the document
totalTime	Total time in minutes used in editing the document
words	Number of words in the document

2.1.5.3 Revision Identifiers in OOXML

In the ECMA 376 standards, the concept of revision identifiers was also introduced. These are 32-bit hexadecimal numbers used to identify the author or author session in which the changes

are made in the document. Revision identifiers were initially introduced to make tracking changes and merging different versions of the same document easier, but since then has been employed actively in forensic investigations and copyright complaints pertaining to office documents.

The identifiers are used to preserve the privacy of the user editing the document. Anonymous editing of a document would not be possible had the standard stored names and timestamps of the edits. The use of identifiers helps protect privacy as well as maintain the integrity of changes made in the document. The office suite provides methods to merge documents that have been forked at some point with a GUI utility.

These revision identifiers are generally found in files containing user data such as *word/document.xml* or *word/settings.xml* amongst other files. Table 2.3 enumerates the seven unique identifiers used by Office Open XML with their use in the four contexts mentioned below. These identifiers can be found within one of the following XML contexts.

- *Paragraph* : In the respective XML file a paragraph is enclosed within `<w:p>`tags with some rich text properties enclosed in `<w:Ppr>`. The actual text is stored in the run tags.

- *Run*: Run encloses a non-block text region which has some common properties. `<w:r>` is used to define a run block. the respective properties are defined using `<w:rPr>` tags and the text is stored within `<w:t>`..

- *Table row*: A table is defined using the `<w:tbl>` blocks of which properties, rows and columns are part of it. Table rows are defines within `<w:tr>` tags. Each of the table row tags have columns (`<w:tc>`), column properties(`<w:tcPr>`), paragraphs(`<w:p>`), runs(`<w:r>`) and text(`<w:t>`) elements.

- *Section properties*: The OOXML standard does not support pages as an element to store paragraph and other content, it instead defines sections. Sections are groupings of paragraphs that share common properties like margin, orientation, size. The section properties are enclosed within `<w:sectPr>` tags.

Table 2.3: Revisison identifiers adapted from ECMA Standard-376 [1] and Didriksen [2]

Name	Context	Description
rsidRDefault	Paragraph	Marks the default id for all run elements without rsidR tag.
rsidRPr	Paragraph	Marks the session in which the glyph character for the target paragraph was last modified.
rsidP	Paragraph	Marks the session in which the properties for the target paragraph was last modified.
rsidDel	Paragraph	Marks the session in which the deletion was performed on target context.
rsidR	Paragraph	Marks the session in which the the associated run context was added.
rsidRPr	Run	Marks the session in which the glyph character for the target run was last modified.
rsidDel	Run	Marks the session in which the deletion was performed on target run context.
rsidR	Run	Marks the session in which the the associated run context was added.
rsidDel	Table row	Marks the session in which the deletion was performed on target Table row context.
rsidR	Table row	Marks the session in which the the associated Table row context was added.
rsidTr	Table row	Marks the session in which the properties for the target row properties were last modified
rsidRPr	Table row	Marks the session in which the glyph character for the target Table row was last modified
rsidDel	Section properties	Marks the session in which the deletion was performed on target context.
rsidR	Section properties	Marks the session in which the the associated Section properties context was added.
rsidRPr	Section properties	Marks the session in which the glyph character for the target Section properties was last modified
rsidSect	Section properties	Marks the session in which section properties were modified
rsidRoot		This tag is found in the settings.xml file, and represents save id associated with the first editing session.
rsid		This tag is found in the settings.xml file, and is a subset of rsidRoot and lists all the revision save id values.

2.1.5.4 Other Identifiers

Relationships As seen in section 2.1.5.1, each document stores relationship files (.rels) amongst other values. These relationship files are used to link other objects that may be present within the file structure. These files are linked with a unique id, path and the respective schema

of the document. These relationships include images, object linking and embedding(OLE) files, controls (scripts to control the content of the documents sometimes ActiveX files), vbaProjects (macros) and hyperlinks amongst others. Table 2.4 refers to these in particular with their respective schema values.

Table 2.4: Identifiers found in various '.rels' files adapted from ECMA Standard - 376 [1]

Element	Schema Value
VBA Project	http://schemas.microsoft.com/office/2006/relationships/vbaProject
Images	http://schemas.openxmlformats.org/officeDocument/2006/relationships/image
Control	http://schemas.openxmlformats.org/officeDocument/2006/relationships/control
OLE objects	http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleobj
Hyperlinks	http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink

Grammar and Spelling Apart from the above the settings.xml file contains `<w:proof-State>` tag to identify if a particular file has been checked for grammar and spelling using w:grammar and w:spelling attributes. Each of these attributes can take two values, 'clean' representing file has been checked, and 'dirty' representing it has not been checked.

Grammar and Spelling identifiers are also embedded within the texts as well. Each error identified by the software is encased in `<w:proofErr>` tags with type attribute representing grammar (gramStart / gramEnd) or spelling (spellStart / spellend) error. Table 2.5 presents a tabular representation of these features.

Table 2.5: Grammar and Spelling Identifiers adapted from ECMA Standard - 376 [1]

Element	Location	Attribute : Attribute Values
w:proofState	settings.xml	w:grammar, w:spelling : clean, dirty
w:proofErr	document.xml, etc	w:type : gramStart, gramEnd, spellStart, spellEnd

2.1.6 Machine Learning Classifiers

Support Vector Machine: Support Vector machines are popular supervised learning method for classification problems. It has been widely studied with the context of malware analysis.

SVM constructs hyperplanes in high dimensional spaces to mark out classification zones. These planes are constructed with the help of support vectors (labelled data) and finding a *maximal* margin where the two categories are distant from each other. Margins are calculated using standard Euclidean distance and are maximised with the help of regression or gradient descent. In the cases where the data is not linearly separable, a kernel is employed to transform the data into a higher dimension where the contents are separable. Radial Bias Function (RBF) is one such kernel. In Lagrangian form a linear SVM can be written as:

$$L_{\mathrm{p}} = 0.5 \cdot \|w\|^{2} - \sum_{i=1}^{l} a_{\mathrm{i}} y_{\mathrm{i}} (x_{\mathrm{i}} \cdot w + b) + \sum_{i=1}^{l} a_{\mathrm{i}} \tag{2.1}$$

where a_{i} represents the Lagrangian parameters, x_{i} are input parameters, y_{i} their corresponding labels+1 or -1. The minima is calculated with respect to parameters of the linear equation $x_{\mathrm{i}} \cdot w + b > 1$, i.e. w and b.

In case linear SVM isn't able to classify samples properly, a transformation function $\varphi(x_{\mathrm{i}})$ is employed to transform the data point to a higher dimension. A kernel is defined as $K(x_{\mathrm{i}}, x_{\mathrm{j}}) = \varphi(x_{\mathrm{i}}) \cdot \varphi(x_{\mathrm{j}})$. RBF is one of such kernels which are widely used in classification problem. It follows the equation :

$$K(x_{\mathrm{i}}, x_{\mathrm{j}}) = exp(-\gamma \|x_{\mathrm{i}} - x_{\mathrm{j}}\|^{2}) \quad for\, \gamma > 0 \tag{2.2}$$

where $\gamma = 1/\sigma^{2}$ and σ is a free parameter.

Random Forests: Random decision forests is an ensemble learning method for classification and regression tasks. They are operated by constructing a large number of decision trees at the time of training. The class attribution is performed by calculating the *mode* solution of all decision trees. A decision tree follows the structure where at every level, including root, a parameter is put forward, and a decision based on this parameter and threshold value is made. In a decision tree based learning method there are a number of metrics that can be chosen like Gini impurity or variance reduction to maximize the classification output. Information Gain shall be used for calculating the optimal parameter thresholds. Decision Trees tend to over fit and lead

to bad results, bagging techniques like random forests by taking a number of trees to solve this issue.

XGBoost: Introduced in [35] and [36] is a ensemble learning method where the learner seeks to create a strong learner in this case from weak learners. eXtreme Gradient Boosting (XGB)sequentially adds predictors and corrects previous models. However, instead of giving weights of different learners with each iteration, XGB fits the new model against the residual values from the previous iteration. The aim is to increase the class representation of weak classes in order to make the classification task, in this case, perform better. By considering residuals, i.e. misclassification iteratively, allows the classifier to perform better.

2.2 Key Related Research

When it comes to detection of malicious office documents, there has been peer-reviewed research published on detection of CFBF formatted documents, while the newer OOXML documents have largely been neglected. This could be attributed to the difficulty of collecting adequate training data – both in number and in quality [37]. Additionally, OOXML is a newer format and has been adopted by malicious authors recently. Visual Basic for Applications (VBA) scripted macros have been a popular attack method in the past, and there have been many attempts towards detection of this particular attack metric. The following section covers the key research produced in the detection of malicious documents. This section discusses the prominent work completed in the detection and analysis of malicious office documents.

2.2.1 Detection of CFBF Documents

In [38], Li et al., authors suggest byte level analysis on an unobfuscated document without any embedded content using n-grams and bloom filters. Methods based on byte level analysis have long been studied and are known to fail against mimicry attacks, polymorphic samples and benign files embedded with malware. In [39], Shafiq et al., a method for detection using Markov n-grams, which shows an increase in detection rate. They tested the Markov n-grams method of

detection on common file types such as DOC, EXE, JPG, MP3, PDF, and ZIP in both encrypted and unencrypted states and reported improvements when compared to previous work. Moreover, it has a higher false positive rate, fails to detect dormant malcode as well as mimicry attacks and polymorphic samples.

In [40], Schreck et al., propose a method to look through malicious files and identify which vulnerability is triggered along with the respective patch or update to the basic application. As part of the methodology, the document is executed in numerous sandboxes running different configuration of MS Office. The method stores logs after the execution of malicious code to identify which vulnerability was triggered. The execution of malware is noted when the Extended Instruction Pointer (EIP) points to an address outside the code segment. This method of detection of both malware and vulnerability fails when challenged with techniques like Return Oriented Programming (ROP).

In [41], Gu et al. propose a method of document classification based on wavelet package analysis. Their method is an amalgamation of entropy analysis and wavelet analysis where each document goes under a wavelet transformation and a feature extraction during the training phase. These features are stored in a library and are used in a feature match process with wavelet features of the unclassified document. They use a smaller dataset of both DOC and PDF files, and thus, findings of this work cannot hold up to scrutiny in a real-world deployment scenario. Moreover, this method has not been studied under various attacks scenarios, and therefore, performance in real-time detection is unknown.

2.2.2 Detection of OOXML Documents

There has been limited peer-reviewed research on the topic of malicious Office Open eXtensible Mark-up Language (OOXML) formatted documents which consider the entire document.

Lagadec [42] discusses the security threats posed by the Office document formats extensively but does not describe any method for generalisation in detection. Naser et al. [43] proposed a method of detection of malicious documents by searching through suspecting keywords such as

o:lock and OLE in the dataset for classification. The dataset used in the testing of this method only contained 57 documents and thus holds little to no statistical significance.

In Lin and Pao [44], the proposed method bears a close resemblance to [45]. The proposed method based on word frequency as features and use support vector machines for the classification of documents. The experiments were conducted on a dataset of malicious and benign PDFs, but the concepts can be easily extended to office document formats such as PPTX and DOCX. They created a simulated dataset to test against mimicry attacks and found a decreased accuracy of 81.38%.

Nissim et al. in their work, [46], suggested an active learning approach using support vector machines. They use the OOXML file structure as a feature for the detector and use structural feature extraction methodology [47] for extraction. They compare their model with five different classifiers J48, Random-Forest, LogitBoost, Logistic Regression and Support Vector Machines to test out their model. Nissim et al. reported a true positive rate (TPR) of 93.6% and a false positive rate (FPR) of 0.19% with an accuracy of 99.6% on a dataset biased towards detection of benign documents. With their active learning approach, they discuss the use of the detector as a supplemental to an active analysis by professionals and anti-virus software. Where an assigned number of documents would be tested by professionals and their signatures would be added to the database for future detection.

However, there are some significant drawbacks of [46]. Primarily, the extraction method only checks for the presence or absence of specific prominent files path and not the contents. Secondarily, the dataset they use is relatively small and contains only 327 malicious files. Additionally, it has been pointed by Miura et al. [48], ALDOCX would fail if the corpus considered had both malicious and benign macros. Further, they also pointed out that the methodology is also susceptible to mislabelling malicious documents that masquerading as benign documents.

Rudd et al. in their work, [49], propose a more generalised method for detecting malicious documents based on file entropy features. They consider over 5 million VirusTotal [21] documents and 1 million documents from CommonCrawl [22] of various extensions both OOXML and CFBF

along with ZIP files. They chose four different byte level features, namely, N-gram Histograms, String Length-Hash Features, Byte Entropy Features and Byte Mean-Standard Deviation Features. Using the previously mentioned features when combined with learning techniques like deep learning neural networks and extreme gradient boosting (XGB) trees led to state-of-the-art results. The gradient boosting method provides worse results as compared to the deep neural network for the document detection problem. The deep neural network is formed using four hidden layers of size 1024, each with rectified linear unit (ReLU) activations. At each layer, they employ a 0.2 ratio dropout and batch normalisation. At the final output, they employ the standard sigmoid cross-entropy loss function. The method they proposed considers the file as a whole and does not consider the nuances by which an attacker might infiltrate. The features used in this methodology are prone to mimicry attacks if used as is, but Rudd et al. hypothesise the use of hashing should make them robust against gradient based attacks, but this has not been experimentally tested.

2.2.3 Detection of Malicious Components

There have been many attempts at detecting a specific type of attacks metric such as macros created with Visual Basic for Applications (VBA) or shellcode. This subsection explores the peer-reviewed research in the detection of malicious macros and shellcode

In [50], Santos and Torres discuss a framework for macro malware detection where the classification algorithm remains interchangeable. They have used python's OleFile and Olevba[32] for extraction of features from 1,671 sample documents. They tested the samples using standard scikit-learn implementations of Neural Networks, Support Vector Machines (SVM), Decision Trees, and Random Forests (RF). They concluded that the best classifier works with an accuracy above 90% and can be used as a good filter for macro malware.

Bearden and Lo [37], implement an n-gram feature selection approach with K nearest neighbour algorithm for detection and classification on p-code present in VBA macros. *p-code* represents the assembly language code generated for the execution of a macro; it is used to provide space-efficient storage and non-version specific execution of the macro. In their experiments, Bearden and

Lo [37] recognise the higher accuracy of 96.3% with 1-gram opcodes compared to 93.4% of 2-gram opcodes.

In [51], Kim et al. discuss the four types of obfuscation techniques that are prominently used. They used five different machine learning techniques, namely SVM, Random Forest, Multilayer Perceptron (MLP), Linear Discriminant Analysis, and Bernoulli Naive Bayes, on 15 features to determine if a document is containing obfuscated macro programming. With these features set, the method achieved the highest accuracy of 97% with MLP and the highest precision of 98.2% with random forests. One should note here; obfuscation detection is not the same as malicious code detection. There is a strong relationship between the two, obfuscation in most cases hints a higher level of scrutiny is required, but it does not prove the maliciousness of the file.

In [48], the authors present a method for the detection of malicious macros using text analysis. The proposed method extracts the macros using Olevba [32], separates the text using various delimiters and replaces the obfuscated text with its category. Using Term Frequency, the number of words recorded are reduced and converted to vectors using Doc2Vec method. The vectors are trained over SVM, RF and MLP and produced the highest F-measure scores of 0.93. [52] uses a similar technique where the authors suggest the extraction of macro first. The macro is used to record features such as Macro Keywords, Count of Integer Variables, Count of String Variables, Shannon Entropy and others. These features are tested against K-Neighbours Classifier, Decision Tree, Random Forest and Gaussian Naive Bayes. The Hyper parameters were searched through using ten-fold cross-validation on a randomised grid search. Two hundred samples were used from each category at the time of training and reported the highest TPR of 98.9% when used with Random Forests.

Snow et al. [53] laid down a technique for detection of shellcode injection attacks using hardware virtualisation. In the method proposed the analyst has the freedom of choosing any heuristic of their choice for the detection process. In the case of malicious documents, ShellOS lets the reader application render the document while ShellOS monitors the memory buffers created for shellcode. The method was tested on PDFs but can easily be translated for Office format

documents. ShellOS used broad-spectrum heuristics and failed when challenged by heap spray attacks, social engineering and shellcode designed to execute under particular conditions.

Iwamoto and Wasaki [54] in their work suggested a dynamic analysis work focused on the extraction of shell code from malicious office document based either on compound binary format or rich text format. In the preprocessing stage, they perform analysis on the file format, disassemble byte sequences and create an order for emulation using entropy. The possible byte sequences are tested and if suspected of having shellcode converted to Microsoft Windows executable for further analysis. Like ShellOS it fails when encountered by return oriented programming attacks, malicious document droppers, and is strongly environmentally dependent. It also fails in the presence of broken samples.

2.2.4 Forensic Analysis of OOXML documents

There has been some work done in identifying the importance of OOXML documents as a source of information in Cyber Forensics.

Fu et. al. in [55], discuss the forensic attributes present in an OOXML document. A document can be expanded and has various files as gathered from Section 2.1.5. The document body is made up of an XML tree in which the Text Element is enclosed within Run Element and Paragraph Element. Each of these elements has unique identifiers assigned to them. The work also discusses the identification process file-copy and content-copy and how embedded metadata can be used for malfeasance.

Additionally, the work also lays out forensics for copyright holders. In the study they note, if there has been any modification done to any one of the files in the document packet, the time stamp of *Last Modified,* remains changed even after repackaging. This can be compared to the *LastModified* metadata stored within the document. Moreover, the optional creator tag can also reveal the author's identity.

In their thesis, [2], Didriksen also discusses these identifiers. Also, they introduce the use of metadata recorded by Office suite as forensic identifiers. They test the validity of this hypothesis

along with if these values could stand to forensic scrutiny. They also tested if the revision identifiers remain constant when copied between documents. They test if the image path stored in the document changes with the version of Office suite as well if data can be ascertained from the thumbnail of the image. They experimentally test the algorithms for assigning RSID values and the probability they might collide when tested in the wild. This thesis uses numerous forensic identifiers discussed in both of these works to differentiate between benign and malicious documents.

2.3 Analysis

It can be noted from Sections 2.1 and 2.2 the attacks fall under two broad categorises. These categories are discussed in detail in Section 2.3.1. Additionally, qualitative analysis is performed on the research work mentioned in Section 2.2. This analysis is presented in Section 2.3.2.

2.3.1 A Broad Classification Of Malicious Document Attacks

All document formats and their corresponding applications provide users with a myriad of features. These features are represented in both the stored document formats and applications. The security gaps in document format and supported application are responsible for the flexibility, complexity and variability of the attacks. They also make effectively analysing and detecting malicious documents troublesome. As described in [56], there are two approaches to the document based attacks.

- Structural attack using crafted content
- Exploitation of programming and interactive application features

2.3.1.1 Structural Attack Using Crafted Content

While making structural attacks, the attackers target the software used to open the document. There are a few popular ones, like Adobe Acrobat Reader for opening PDFs Microsoft Office suite for Rich Text Documents, Presentations and Spreadsheets. With the knowledge of the software,

the attacker steers structural attacks to cause an unexpected reaction from the system. With this opportunity, the attacker can embed malicious code in the memory for later execution. One of the conventional methods of these attacks is memory overflow. The attacker craft documents to make a vulnerable value overflow. Return Oriented Programming (ROP) attacks are frequent amongst structural attacks.

Example: A text field could be a megabyte large without any logical reason and may contain malware code. A poorly written software could try to read the whole field into a smaller memory location, causing a buffer overflow and leading to heap spray.

2.3.1.2 Exploitation Of Programming And Interactive Application Features.

Unlike the previous approach, instead of attacking the software, the attacker targets the features provided by the application format itself. As previously mentioned, each document format provides a user with features like embedding images, videos and sometimes scripts to facilitate work. Attackers use these interactive application features like embedding flash animations, ActionScripts or visual elements like video or audio, and programming features, like writing JavaScripts, Macros or Visual Basic Actions, to load and execute malware.

Example: An attacker can create a document which uses the Visual Basic functions AutoOpen and Shell to launch a malicious shell script that was stored in the document. This launch behaviour can be set up on various triggers and not just opening of the file.

2.3.2 Qualitative Analysis

Both the methodologies discussed in [38], and [39] have concentrate on the detection of malicious office documents following CFBF formatted documents. These techniques only consider unobfuscated documents and use byte-level statistics based features. The proposed features can be used to represent any file-type and not just Office documents. As already mentioned, this feature representation can be bypassed with ease with a polymorphic attack. Methodology in [41] uses wavelet analysis for the detection and has similar drawbacks as the other two methods. Also,

the authors used a small set of PDF and DOC files and did not test the method under realistic deployment scenario.

The methodology suggested in [40] focuses on detecting which patch of the Office suite was targeted by the vulnerability. The detection method proposed here is weak and only takes into account malcode that is present in a selected memory region. By targeting only a section, A widespread attack, where the attacker triggers a shared vulnerability to heap spray malcode in the memory and later triggers its execution, is left unaccounted. The detector and path identification system only considers attacks which either have shellcode as part or use shellcode triggers. These constraints limit the number of malware the method can successfully detect.

ALDOCX [46] uses machine learning and structural features for the detection of malicious documents. While testing the authors only used 327 malicious files on a dataset biased towards benign files. The method also uses just the surface level features, i.e. whether a file or a link to a file present or not but does not look at the particular contents. Through experimentation, they arrived as SVM as the classifier which has known to fail against adversarial attack [57]. Additionally, it has been pointed by Miura et al. [48], ALDOCX would fail if the corpus considered had both malicious and benign macros. Further, they also pointed out that the methodology is also susceptible to mislabelling malicious documents that masquerading as benign documents.

The method proposed by [49] uses 6 million documents of various file formats to train and test deep neural networks and XGB. The method proposed here uses format agnostic features and limited domain knowledge to define features. As discussed in Chapter 1, the malware industry is shifting form packed malware to customised and staged execution. In a staged execution, only a small section of malware is stored with the document, upon execution the malcode downloads sections and may trigger vulnerabilities present in other software. As these are shorter code segments, they may not get recognised with the other agnostic features. The method uses several byte features which may be time-consuming to extract, and it cannot be customised to detect only a certain malware or some new domain-specific vulnerability.

Detection of malicious macros has been studied heavily, and several techniques have been

Table 2.6: Summary of the reviewed Office Document detectors.

Detector	File Formatted	Static or Dynamic	ML?
[38]	Detection of CFBF Documents	Static	No
[39]	Detection of CFBF Documents	Static	No
[53]	Detection of Malicious Components	Dynamic	Yes
[40]	Detection of CFBF Documents	Dynamic	No
[44]	Detection of OOXML Documents	Static	Yes
[41]	Detection of CFBF Documents	Static	No
[54]	Detection of Malicious Components	Dynamic	No
[46]	Detection of OOXML Documents	Static	Yes
[50]	Detection of Malicious Components	Static	Yes
[37]	Detection of Malicious Components	Static	Yes
[49]	Detection of OOXML Documents	Static	Yes

developed in the past. Methods [50], [37], [51], and [48] all use a combination of domain-specific features and machine learning models. Each of these detection methods focus on one aspect of the malicious document problem and thus fail to gather the global picture. Method [48] reports an F-Score of 0.93 on a very small dataset. [50] an accuracy of only 90% which is probably better than most antivirus software. [51] is the only method that reports an accuracy of more than 97%.

Methods [53] and [54] were employed for detection of malicious shellcode in documents. [53] proposed an analysis method which focuses on the memory block used in the execution. It was tested with broad heuristics on only PDF files. It failed when challenged with techniques like heap spray and ROP. [54] is a dynamic analysis method which depends on the environment heavily. Once the shellcode was extracted from the CFBF or RTF file, it analyses under dynamic conditions. Like [53] it failed against heap spray and ROP techniques.

Most techniques that use machine learning for the detection of malware assume all possible malware configuration is known at the time of training, whereas it is known not to be the case. Malware samples and attack techniques mutate rapidly and pose a severe threat to the end user. Additionally, the metrics used by the detectors overestimate their performance in a real deployment scenario.

2.4 Research Gaps

This section highlights the literature gaps identified by the analysis of Section 2.2.

R1: There has been limited peer-reviewed research that deals with the detection of malicious Office in either format. Researchers have focused on how to segregate malicious components, like shellcode and macros, as seen in Section 2.2. It was also noted in Section 2.3 the detectors which tried to classify components reported worse results in performance metrics. When it comes to the OOXML format, there have only been two significant peer-reviewed articles that use the entire document. [46] is biased and only focuses on structural paths present in a file, where as [49] uses format agnostic features for detection. There has been no research conducted which takes into account the domain knowledge of the OOXML while performing the detection task.

R2: As it was already noted in Section 1.1 malicious actors have shifted from a more generalised approach to specific attacks. These attacks use spear phishing to entice the user to execute their malware. Additionally, malware deployment now uses a multi-stage tact, where the malware is downloaded in stages from the attacker's server. This method of attack focuses on exploiting vulnerabilities in the software. Most detectors, as previously discussed, use a rigid feature set which may not detect such novel attacks. No research has been perused, which focuses on making the detector customisable at the time of deployment.

R3: Except for a few approaches like [49], detectors which use machine learning algorithms have made the presumption all malicious samples and families are known at the time of training. Whereas it has been noted in [4], the rate of creation of new malware and attack technique is unprecedented. This form of training and testing provides unrealistic estimates of detector performance. Temporal evaluation, where the dataset is split according to some time metric, provides a more realistic evaluation. This method of evaluation has not been pursued except for a few research articles.

R4: The datasets used in some of the research articles discussed use smaller or simulated datasets. The dataset should also be prepared in order to reduce the maximum classification

bias. The use of smaller or simulated datasets for detection problem raises a question about the generalisability of the method proposed. The use of these datasets should be limited to the creation of proofs-of-concept as the performance metrics used for evaluation present skewed and unstable results.

2.5 Problem Formulation

Based on Sections 2.2, 2.3 and 2.4, it is clearly evident the domain of detection of malicious OOXML is sparse. There are no methods available that consider the entire document and use domain knowledge of OOXML for the detection of malicious documents.

To address the aforementioned research gaps, a malicious OOXML detection methodology is required, which makes use of features based on the entire document and the domain knowledge OOXML schema. These features need to be customisable at deployment if the need arises. The methodology should use machine learning in order to generalise the knowledge of novel attacks. The dataset in consideration should comprise of samples collected from the everyday systems and not simulated. Both standard and temporal evaluations should be performed to ascertain the overall performance of the system. Additionally, the aim is to maximise Accuracy, TPR, AUC, Precision, Recall and F-Score and minimise the FPR.

2.5.1 Evaluation Metrics

The classification problem is a supervised learning problem, i.e. for each sample in the dataset the actual label is known. At the time of classification, a predicted class label is assigned to the sample. This label may or may not be the same as the original label. For instance, if a file is assigned the malicious label after classification where as the original label was benign. This is a case of misclassification. A confusion matrix for binary classes is a 2×2 represents of all possible outcomes in the form of TP, FP, FN, and TN. Table 2.7 presents the four cases in a tabular form.

There were six metrics used to measure the performance of the detector system namely, accuracy, false positive rate (FPR), precision, recall or True Positive Rate (TPR), F-Score and

Table 2.7: Confusion matrix.

		Actual class	
		Label 1	Label 2
Predicted Class	Label 1	True Positive (TP)	False Negative (FN)
	Label 2	False Positive (FP)	True Negative (FN)

Area Under the ROC Curve (AUC). Accuracy measures the total error observed during the testing phase of the detection. Equation 2.3 represents the mathematical equation to calculate accuracy. FPR measures the classifiers ability to identify samples with an actual negative label as positive. FPR is calculated by taking a ratio of false positives overall negatives (See Equation 2.4). Recall or TPR measures the ability of the classifier to recognise the positive samples as positive correctly. It is the ratio of TP samples over all samples which were labelled as positive at the time of testing (See Equation 2.5). Precision measures the correctness of the classifier. The metric is a ratio between predicted positive samples with all samples that were predicted positive (See Equation 2.6). F-Score measures the accuracy of the tests. It is calculated by taking the harmonic mean between precision and recall (See Equation 2.7). This metric was used to compare the results of the analysis with previous work.

$$Accuracy = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{2.3}$$

$$FPR = \frac{FP}{(FP + TN)} \tag{2.4}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{2.5}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{2.6}$$

$$F - Score = 2 * \left(\frac{Precison * Recall}{Precison + Recall} \right) \tag{2.7}$$

2.6 Summary

In this chapter, a through background for the Office malware problem is provided. This included a discussion on the office file formats, malicious toolkits, and analysis techniques. The body of literature about the detection of malicious office documents is summarised and discussed. Analysis of the literature was completed, and research gaps were identified. The problem tackled throughout this thesis is formulated, and a performance metric was introduced.

Chapter 3

Methodology

This section introduces the proposed solution along with the hypothesis and analytical validation of it. Two hypotheses are put forward, based on which the methodology is established. These hypotheses are validated using a small cross-section of samples.

3.1 Proposed Hypothesis

The following hypothesis are made while designing the detection:

(1) **H1:** It is hypothesised forensic identifiers and metadata present in an OOXML document are ideal candidates for customisable and domain-specific feature representation. Additional representation in the form of structural paths and entropy of selected files, as well as byte histograms entropy and byte histograms, provide robustness to the feature domain.

(2) **H2:** It is hypothesized that metadata timestamps collected from XML tags like *created*, *lastPrinted*, and *modified* can be effectively used to split the dataset for temporal analysis.

3.2 Proposed Solution

The proposed solution is static extraction, machine learning-based system with the goal of distinguishing between malicious and benign Office documents. It leverages the file structure of an Office document along with the stored metadata and some platform agnostic features. The goal was to use a combination of previously discovered state-of-the-art features (See Section 2.2) along

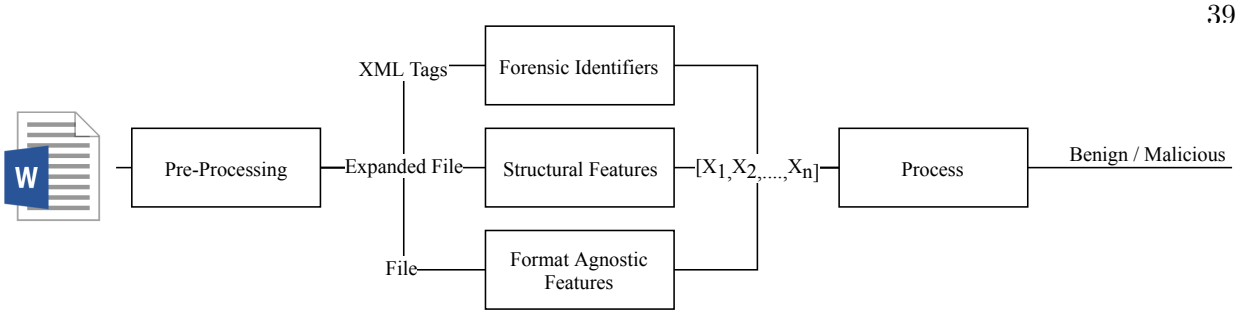


Figure 3.1: Architecture of the proposed solution

with forensic features discussed in [2] to achieve an efficient detection of Office malware. Figure 3.1 gives an overall architecture of the system, which can be divided into the following three key modules.

(1) **Parser** Like it is the case with other document based file formats, PDF, in particular, this module first validates the file in question according to the ECMA standard then extracts structure, embedded forensic markers and entropy features from the file.

(2) **Feature Extraction** This module takes the information extracted by the parser as the input and converts it into vectors characteristics to depict the file.

This module decides based on the vector characteristics if the current sample under inspection in malicious. This is a mathematical function which decides on appropriate parameter values based on the training set. Once the parameters have been fixed, the classifier can identify between benign and malicious files which have not been included in the training set.

3.2.1 Parser

The parser module inspects whether or not the document is following the ECMA standards using the Open XML Software Development Kit(SDK) 2.5 for Office, which was first introduced in 2017. The SDK uses System.IO.Packaging APIs and .NET framework, which simplifies the task of examination. Since there is not a standard library for parsing and extraction of OOXML

documents, like with the case of [58], parser and extractor were written in python. The parser is responsible for the following tasks:

1) Validate if the file in question follows the proper schema. 2) Expand the Office file and its sub-components, like macros 3) Extract the structural, embedded forensic markers, and entropy features 4) Validate the file has not been tampered, thus protecting against mimicry and reverse mimicry attacks.

3.2.2 Feature Extraction

The feature extraction process is the fundamental component of the entire system, as without optimised feature the detector may underperform. It converts the information collected by the parser along with the original package itself and converts it into features for the malware classifier. The features extracted from the OOXML file can be sub-classified into the following three categories, a) format agnostic features, b) structural features and c) forensic features. Figure 3.2 outlines the features extraction process, along with all its sub-processes and functioning.

3.2.2.1 Format Agnostic Features

These are features that do not require any prior knowledge of the format or the domain and thus can be adapted for malicious detection problem. The features in [49] were modelled from this concept. In the labelled dataset [59], apart from the standard features, they also provide some format agnostic features. These features were extended to the malicious document problem.

Byte Histogram: These features are calculated by considering the file as one unit. The file is explored byte-wise, and a histogram is created with the 256 possible values represented on one axis and their frequency on the other with 32 bins. These histograms are then normalised against the file size, which is collected as part of the structural features. Figure 3.3 plots normalised byte histograms between two documents, one from each category. As can be seen from the figure, there is a disparity between the frequency of symbols which is used by the classification algorithm.

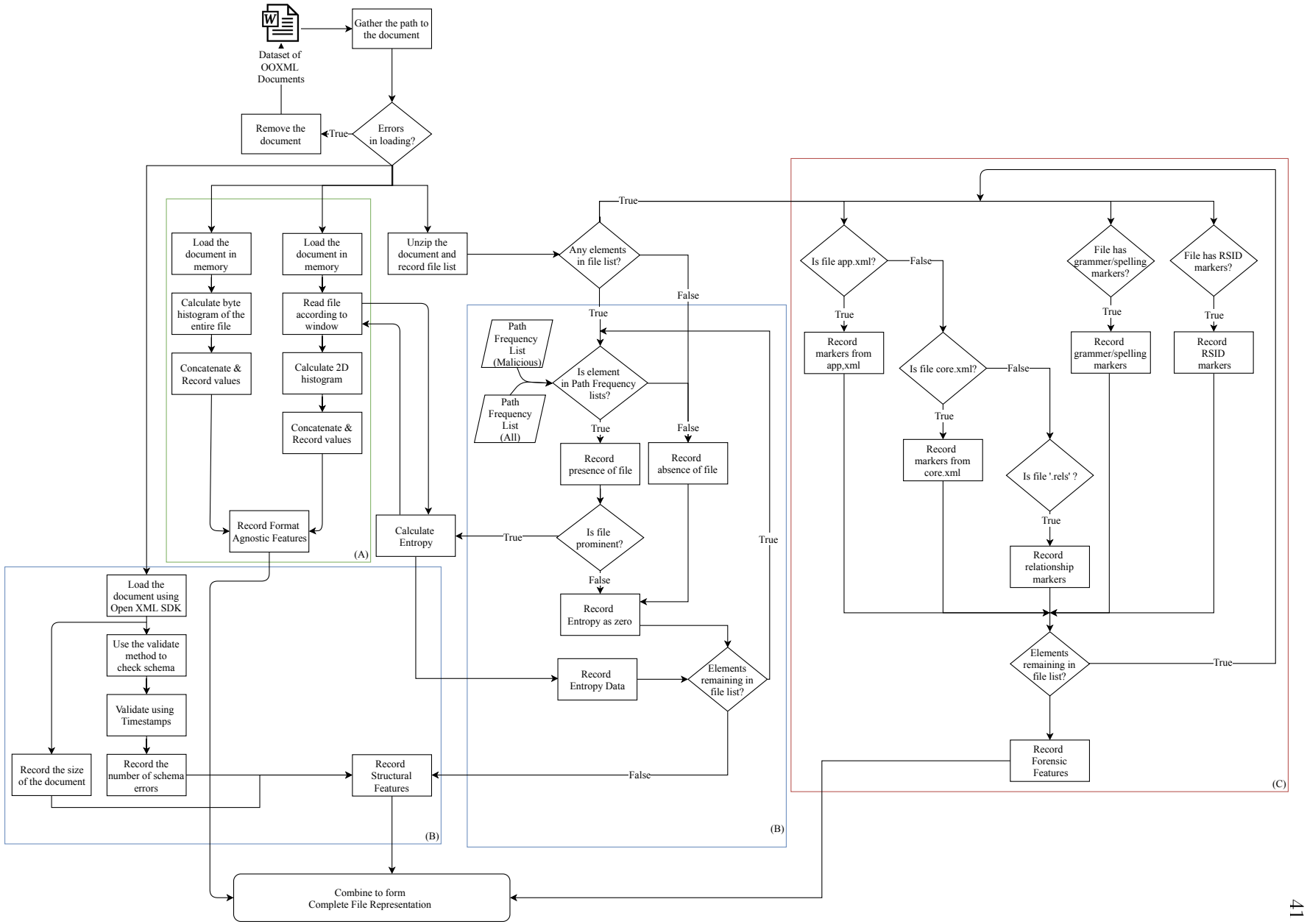


Figure 3.2: Working of the extractor. (A) Format Agnostic Features (B) Structural Features (C) Forensic Features

Byte-entropy histogram: They were initially introduced by Saxe et al. [60] in their research of detecting malicious binaries using deep learning and two-dimensional features. These are obtained by taking a fixed-size sliding window (1024 bytes) over strides of 256 bytes and a bin size of 16 x 16 bins.

$$S = -\sum_{i=1}^{256} p_{\mathrm{i}} \cdot log_2(p_{\mathrm{i}}) \tag{3.1}$$

With each stride, entropy is calculated for each byte value using Shannon's Entropy with base 2 (Equation 3.1), where p_{i} is the i^{th} proportion of the window. A two-dimensional histogram is taken over byte value and entropy and concatenated to form a feature vector similar to, as mentioned in [60]. Figure 3.4a and 3.4b show the 2D byte entropy histogram of two sample document. As can be inferred from these two figures, the entropy for a malicious file is varied whereas for the case of benign document these values are high and focused in the 7-8 value range, this disparity in the values used by the classification algorithm to distinguish between the two sets. Table 3.1 provides a summary of the format agnostic features discussed in this section.

Table 3.1: Summary of Format Agnostic Features

Feature	Description	Data Type
Byte-entropy histogram	2D byte entropy histogram of the file	256 floats
Byte histogram	Byte histogram of the file	32 floats

3.2.2.2 Structural Features

The second set of features are derived from the document and the metadata stored by the operating system. As seen in Section 2.1.5, a OOXML follows a strict hierarchical structure. Using this structure, each file is expanded into the relevant sub-files which are later used to represent the structure. With each such extraction, we get more than 22 files and directories. Most of these files are of XML type and store user information in the document.

In Nissim et al. [46] the authors chose to use the structural path along with path identifiers

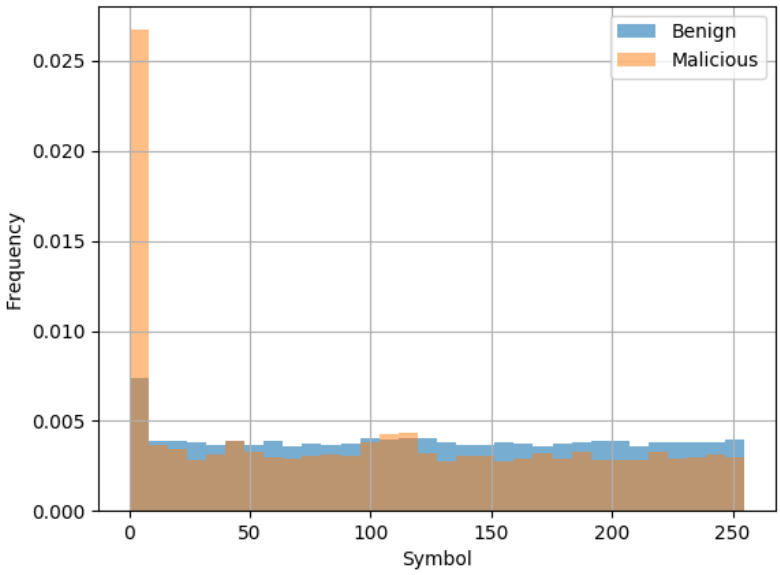
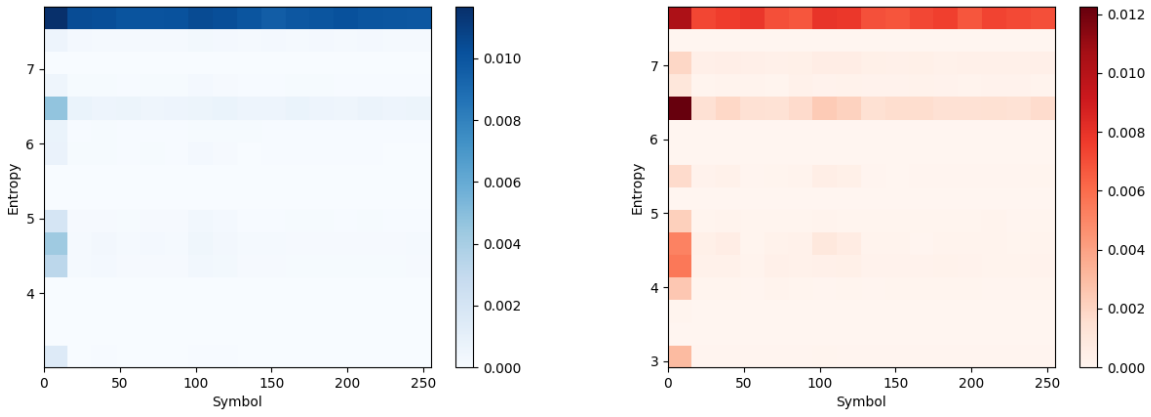


Figure 3.3: Normalised Byte Histogram Comparison between One Documents of Each Category



(a) A benign document

(b) A malicious document

Figure 3.4: Byte Entropy Histogram of two randomly chosen documents from each category

Table 3.2: Summary of Strutural Features

Feature	Description	Data Type
Size	Stores the size of the file in number of bytes	integer
Error Number	Number of errors found by Open XML SDK	integer
Prominent files	Presence or absence of file based of factor	boolean
Entropy of prominent files	Entropy of prominent Files	float

from the XML as a feature. This method of extraction is rather brutish and leads to redundant features which provide no additional actionable information. The solution adapted the best of their features and extracted just the directory tree from each of the document. Parser module validates each file for indicators of tampering or alteration.

Using the ValidateWordDocument API provided by Microsoft Open XML SDK 2.0 [61] was used to validate the document and count the number of errors found in each document in the set. The size and extension of the file under inspection were also stored, and all three of these values were modified into features.

Besides, file entropy of these prominent files is also recorded and converted to a feature. A file is deemed prominent if it lies between the upper and lower limit or if the file is present in malicious labelled documents. Shannon's entropy with base-2 is applied here as well to calculate the entropy. These values are stored in the CSV file and converted into a feature for a machine learning algorithm. Table 3.2 presents a summary of all structural features while Table 3.3 enumerates the 7 structural features and their respective importance scores calculated using random forests on the temporal dataset mentioned in Section 4.2.1

3.2.2.3 Forensic Identifiers

Unlike the other categories, forensic identifiers require a domain understanding of the file format. The quality of these features is highly depended on the parser in use. As Didriksen [2] mentions, utilities like FTK and EnCase are not able to capture all of the forensic markers discussed earlier. Moreover, the parsing process is a prime target where an attacker can interfere with getting

Table 3.3: Indicative Prominent Structural Features with respective Type and Feature Importance Score

Feature	Type	Feature Importance
\word\vbaData.xml	Boolean	0.05170384882297021
\word\vbaProject\ThisDocument.cls-entropy	Boolean	0.05113221614376270
\word\vbaProject\ThisDocument.cls	Boolean	0.04689803294395375
\word_rels\vbaProject.bin.rels	Boolean	0.04194805599678933
\word_rels\vbaProject.bin.rels-entropy	Boolean	0.03852556121711357
\word\vbaProject\Module1.bas-entropy	Boolean	0.03706039471774040
\[Content_Types].xml-entropy	Boolean	0.02754314563313486

a likely outcome. In the case of our solution, the eXtensible Markup Language (XML) Document Object Module (DOM) python libraries were used to parse XML files for metadata and forensic identifiers. As can be inferred from Sections 2.1.5.2, 2.1.5.3 and 2.1.5.4, a OOXML document editor captures hundreds of unique tags in the document schema. They serve numerous roles from maintaining the document integrity to allowing users to merge and track various versions of the document.

Historically, cyber forensics has been used to identify, recover and trace effects of any attack. With malware, forensics is primarily used to trace the steps taken by a malicious actor. Most malicious campaigns are mostly automated and have minimal human input. Besides, attackers avoid detection by explicitly removing the identification markers or by forging them, whereas software developers and users are encouraged to share their identity and certificates. Malware also employs obfuscation to avoid detection from anti-viruses and common detectors. The absence of common identification pointers and the presence of obfuscation creates a differentiation between the software categorised as malicious and benign. Table 3.5, describes some of the prominent forensic features along with their respective importance scores calculated using random forests on the temporal dataset mentioned in Section 4.2.1

The OOXML tags like *Application, Creator, Company, LastModifiedBy, Template*, provide information about the user identity. The revision identifier tags that were discussed in Section 2.1.5.3 are key to identifying what changes were made. The revision identifiers are embedded with

Table 3.4: Summary of Forensic Identifiers

Feature	Description	Data Type
Revision Identifiers	Identifiers as discussed earlier in Section 2.1.5.3	18 integers
Relationship Identifiers	Identifiers as discussed earlier in Section 2.1.5.4	5 integers
Grammar and Spelling	Identifiers as discussed earlier in Section 2.1.5.4	3 integers
app.xml indicators	Identifiers as discussed earlier in Section 2.1.5.2	24 integers
core.xml indicators	Identifiers as discussed earlier in Section 2.1.5.2	14 integers
timeSincePrinted	Time difference between the time at extraction and last printed (in Seconds)	integer
timeSinceCreation	Time difference between the time at extraction and created (in Seconds)	integer
timeSinceModified	Time difference between the time at extraction and last modified (in Seconds)	integer
timeSpentInDocument	Time difference between last-modified and created (in Seconds)	integer

every change in content or setting is performed, and these values increase from the *RsidRDefault*. If there are any changes made in the document through an application that supports ECMA-376, it would be reflected in the 'rsid' values. Additionally, if a document was created by people for communication, more often than not, these documents are edited more than once. *Unique Rsid values, revision, TotalTime, pages, words, lines, characters* are also indicators user activity in the document. The OOXML standard records the following three timestamps, *Created, LastModified and LastPrinted*. These timestamps are indicators of user behaviour and such used for identifying discrepancies in the document. Furthermore, the standard has the option to record *Language, Grammatical and Spelling mistakes, title, words*. These indicators would either be absent or will be present in small number in malicious documents, while benign documents have these indicators in plenty. As seen in Table 3.5, *Words, countVBA* and *CharactersWithSpaces* are some such features which hold high importance. The various revision identifiers, *PrsidRDefault, PrsidRPr, RrsidRPr* and *PrsidP*, primarily represent the presence of these indicators in large numbers in benign and in dwindling numbers in malicious documents. Table 3.4 summarises all the forensic identifiers discussed.

Table 3.5: Indicative Prominent Forensic Identifiers with respective Type and Feature Importance Scores

Feature	Type	Feature Importance
Words	Integer	0.03912780051174833
countVBA	Integer	0.03398039378328111
PrsidRDefault	Integer	0.02920306337860408
PrsidRPr	Integer	0.02409092960167331
CharactersWithSpaces	Integer	0.01781379900371534
RrsidRPr	Integer	0.01739624013097420
PrsidP	Integer	0.01597941252636012

3.2.3 Feature Selection Techniques

Feature selection plays a very crucial role in the differentiation between unknown malware and benign files. Most feature selection techniques use a filter based approach, in which they gauge the importance of each feature in relation to the class variable (i.e. malicious or benign) while performing a prediction task on an unknown sample. The features we have mentioned in the previous section lead to high dimensionality at the time of extraction, feature selection also allows for a reduction of this dimensionality as well as faster training of the machine learning model. It should also be noted, regularisation and feature selection has lead to better accuracy in classification problems. This work experiments with the following three feature selection techniques:

3.2.3.1 Chi-Square (CS)

The Chi-Square test [62] is another popular feature selection technique and is used to measure the dependence between two variables, in this case, feature N_i and class C_k. A high score represents a close relationship between two variables. The Chi-square formula is as follows:

$$\chi^2(N_i, C_k) = \frac{N|PS - QR|^2}{(P+R)(Q+S)(P+Q)(R+S)} \tag{3.2}$$

where N is the total number of features in the file, P is the number of features present in C_k containing N_i, R is the number of features not present in C_k containing N_i, Q is the number of features in C_k not containing N_i and the S is number of features not in C_k and not containing N_i.

3.2.3.2 Information Gain (IG)

Information Gain (IG) [63] is commonly used in feature selection method in tasks related malware classification. For a given i^{th} feature(N_i) the IG score is calculated using :

$$IG(N_i, C_k) = \sum_{N_i=0}^{1} \sum_{C_k \epsilon (B,M)} P(N_i, C_k) \log \frac{P(N_i, C_k)}{P(N_i)P(C_k)} \tag{3.3}$$

where C_k represents the two available classes, malicious or benign and N_i represents the presence or absence of a particular feature in a file. $P(N_i,C_k)$ is the proportion where feature N_i is present in the files belonging to class C_k. $P(N_i)$ represents the ratio of files with feature N_i) to total number of files and $P(C_k)$ represents the ratio of class C_k to the total number of classes.

3.2.3.3 Feature Importance (FI)

The Scikit-Learn in python library allows the use of feature importance as a selection technique, where the importance value is the relative depth of a feature as a decision node in a tree. The features at the top of the tree contribute significantly more while making predictions as compared to features at the lower nodes. These expected fraction values can be thus used as a measure of the relative importance of features. The feature importance scores in Tables 3.3 and 3.5 were calculated using random forests model on a dataset containing 2300 samples with 28% malicious content.

3.2.4 Classification

The features extracted with the proposed method can be used with different classification algorithms. In Chapter 4, this statement was verified by testing the proposed model features against different classification algorithm. In particular, Linear Support Vector Machine (SVM), SVM with Radial Bias Function (RBF) kernel, Random Forests and eXtreme Gradient Boosted trees were employed as these have historically given significant results in the malware detection task.

3.3 Analytical validation

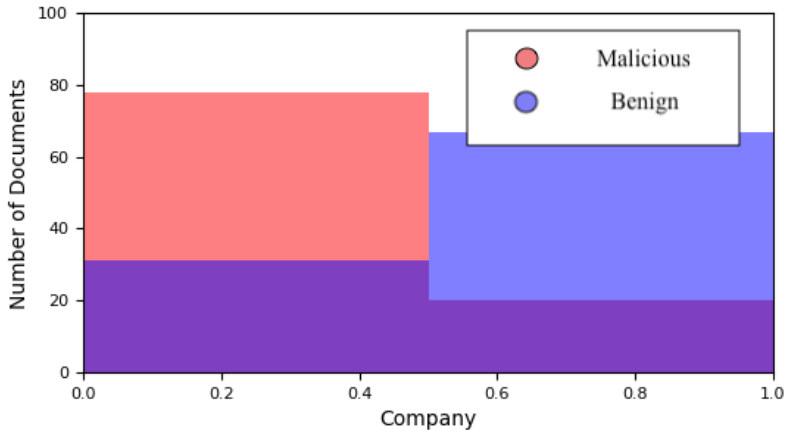
To validate each hypothesis, a set of 99 malicious documents and 50 benign documents with *DOCM, DOTM* extensions and 49 benign documents with *DOCX, DOTX* extensions were sampled. Forensic identifiers mentioned in the Section 3.2.2.3 were extracted from these files.

3.3.1 Validation of Hypothesis 1

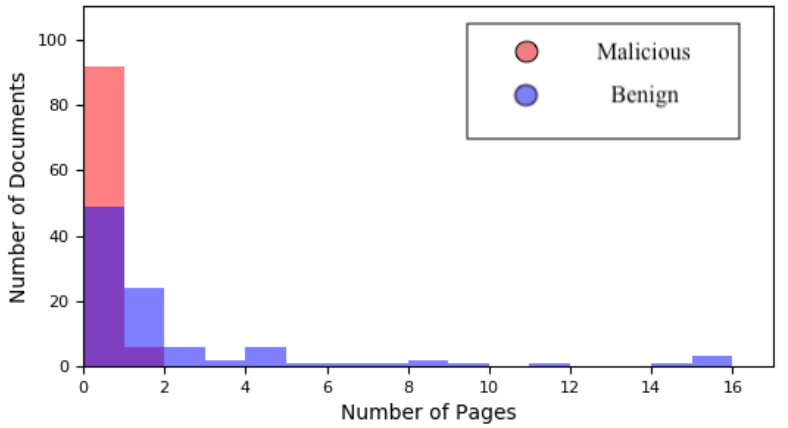
Hypothesis 1: It is hypothesised forensic identifiers and metadata present in an OOXML document are ideal candidates for customisable and domain-specific feature representation. Additional representation in the form of structural paths and entropy of selected files, as well as byte histograms entropy and byte histograms, provide robustness to the feature domain.

Analysis At the core of the Hypothesis-1 is the assumption that forensic identifiers are uniquely positioned to quantifiable measure a difference between the two classes. This assumption is tested by extracting the identifiers mentioned in Table 3.4. These identifiers are plotted as over lapping histograms as can be seen by Figures 3.5 and 3.6. Figure 3.5a represents the binary value recorded at the time of extraction. If the *Company* tag is present the value is recorded as *1* else it is recorded as *0*. The histogram is created with two bins, and as it is apparent, most malicious documents about 80% of the selected do not record this value while about 75% of benign documents store it. Similarly, the metadata tag *Pages* was recorded from this test dataset. It can be noted from the histogram which was created using 16 bins in Figure 3.5b, most malicious documents do not use a small number of pages while the same cannot be said about the benign set.

Figure 3.6 extends the this evaluation to *RSID* and *Words* tags. The total number of RSID values calculated by taking the sum over the various RSID values discussed in Table 2.3. This count is calculated for both malicious and benign documents. Higher values indicate higher levels of activity and revisions by the user in the document. As can be seen from Figure 3.6a created using 30 bins, over 80% of malicious documents show little to no revision activity, whereas this is not true for benign documents. Similar to the *Pages*, the *Words* tag is also a measure of activity



(a) Whether the company tag has been recorded in metadata

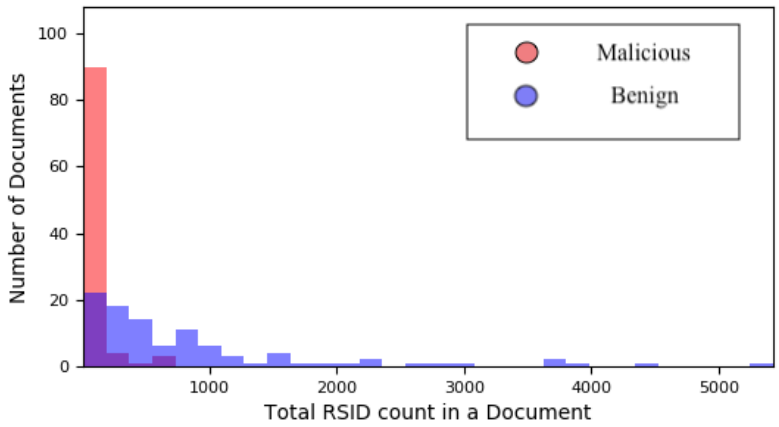


(b) The number of pages per document recorded in the metadata

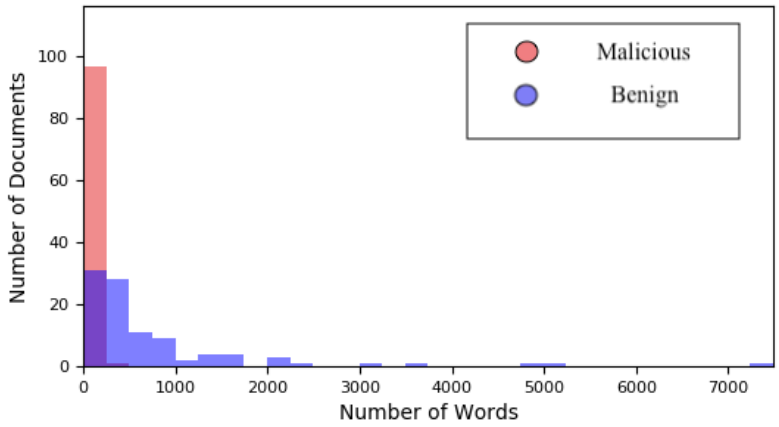
Figure 3.5: Comparison via histograms for some metadata values extracted from 99 malicious and 99 benign documents.

performed by the user. The total number of words written in a document is recorded as metadata and stored in the *app.xml* file. Figure 3.6b, a histogram of *Words* tag with 30 bins, clearly shows a distinction between the malicious and benign documents. There is a clear lack of any words written in a malicious document. This lack of activity, when compared to the other counterpart, is a clear distinguishing feature.

Based on the data collected and the proof-of-concept presented in this subsection, it is evident that forensic identifiers and metadata collected from the document would serve as a good metric



(a) Total number of RSID per document recorded in the metadata



(b) The number of words per document recorded in the metadata

Figure 3.6: Comparison via histograms for some metadata values extracted from 99 malicious and 99 benign documents.

to distinguish between the two sets of documents. Features similar to those discussed in Sections 3.2.2.1 and 3.2.2.2 have been used individually to represent malicious samples in [59] and [46] respectively. By adding these representations, a classifier is afforded more data points to make the classification. Since, the forensic and metadata features are limited to a strict schema, these feature representations cover novel methods of attack which may not have to conform to the strict schema. Additionally, format agnostic features, byte-entropy and byte-histogram-entropy cover the entire document in the packed form thereby covering the file in the entirety. While the structural

features, take into account common paths and entropy of select files, thereby covering the common attacks and their origin points. As it can be interpreted from the above arguments, adding these features leads to a more secure feature representation.

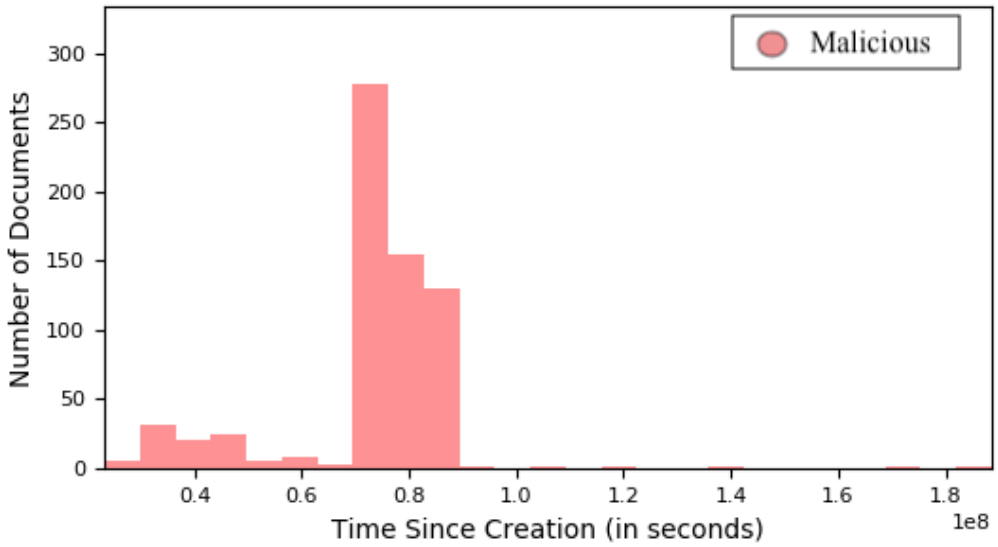


Figure 3.7: Histogram representation for *timeSinceCreation* metric for malicious documents.

3.3.2 Validation of Hypothesis 2

Hypothesis 2: It is hypothesized that metadata timestamps collected from XML tags like *created*, *lastPrinted*, and *modified* can be effectively used to split the dataset for temporal analysis.

Analysis The metadata tags *created*, *lastPrinted*, and *modified* record the timestamps of creation, last printing and last modification respectively. These timestamps can be converted into more discerning features. Tags like *timeSinceCreation, timeSincePrinted, timeSinceModified, and timeSpentInDocument* can be calculated using the time of extraction as a reference point as shown in Table 3.4. In order to validate the hypothesis, the *created* tag from 664 malicious documents was extracted and converted using *timeSinceCreation*. The malicious documents were collected using the VirusTotal service under their academic licensing. Histogram was calculated over *timeSinceCreation* using 25 bins and is presented in Figure 3.7. As it can be ascertained from

the figure, a large number of documents in the corpus were created over 7×10^7 seconds ago. Based on Figure 3.7 and raw data, 1 January 2018 was selected to split the document corpus into training and testing dataset for temporal evaluation.

3.4 Summary

In this chapter, two hypotheses were presented and validated using data collected from the document corpus. Structural, agnostic and forensic identifier features were presented. An extraction methodology was put forward to extract the three categorises of features. Three feature selection techniques and machine learning algorithms are discussed, as well. The hypotheses were validated with features collected from 200 documents as well as a corpus of malicious documents.

Chapter 4

Experiments and Results

This section discusses the four experiments in order to validate the proposed hypothesis. The base detection rate of consumer antivirus software is established. The standard evaluation is performed to test the performance of the method in different configurations. The performance of the method is also tested against a temporal set to establish generalisation capabilities. The findings and discussion from these experiments are also presented.

4.1 Evaluation of Malicious OOXML Documents Against Existing Antivirus

With the help of academic licensing from VirusTotal, MS Office documents were collected. These documents were sorted by their file extensions, and the data is recorded in Table 4.1. These files provide a unique opportunity to understand how malware is currently getting developed and study past trends.

4.1.1 Experiment Description

To get a preliminary analysis and base recordings of how the latest antivirus software respond to the malicious OOXML documents, an experiment is conducted in an air-gapped virtual machine. The experiments take the files downloaded using the VirusTotal service and load them into a virtual machine running Windows 10. The default antivirus software, Windows Defender, was updated to the latest definitions. To maintain an unbiased perspective, Kaspersky Total Security was also installed on the machine. Custom scans were executed on the OOXML document set containing

Table 4.1: Number of documents collected from VirusTotal

Document Format	Malicious
OOXML Document	705
CBF Document	812
OOXML presentation	19
CBF presentation	11
OOXML Spreadsheet	78
CBF Spreadsheet	589

(DOCX, DOCM, DOTM, DOTX) extensions.

4.1.2 Result and Discussion

The collection of malicious OOXML Documents was tested against the updated definitions of Windows Defender and Kaspersky Total Security. The results of this evaluation are presented in Table 4.2.

As it is visible from the above experiment, the antivirus software available to the average consumer is ineffective against the problem of malicious documents. Windows Defender only reported 30 files as malicious from a possible 705, i.e. only 4.25%. Most users do not upgrade their base protection provided by the operating system with the hopes they are protected against most malware attacks. Antivirus provider, Kaspersky, records only 109 files, which is a mere 16.45%.

4.1.3 Conclusion

This experiment makes it quite evident; the current consumer malware detection software is ineffective in detecting malware. Therefore, there is a requirement for specialised software which can detect such files.

Table 4.2: Results from testing 705 malicious documents against Windows Defender and Kaspersky Total Security

Document Format	Total	Windows Defender	Kaspersky Total Security
OOXML Document	705	4.25% (30 files)	16.45% (109 files)

Table 4.3: The composition of Datasets for standard evaluation.

Name	Malicious	*.docm* and *.dotm*	*.docx* and *.dotx*	Malicious : Benign Ratio
Set I	664	552	444	40 - 60
Set II	664	552	997	30 - 70
Set III	664	552	2,104	20 - 80
Set IV	664	552	12,076	05 - 95

4.2 Standard Evaluation of the Proposed Solution

Here the proposed method was trained against randomly chosen WordProcessing files from the malicious and benign datasets. Once trained, it was tested against the previously unseen set of files. This evaluation provides us with general performance in terms of false positive rate (FPR) and true positive rates (TPR).

4.2.1 Dataset

Using the CommonCrawl APIs [22] 63,000 OOXML Word formatted *(.docx, .docm, .dotx, .dotm)* were downloaded. Through the same method, 552 *.docm* and *.dotm* formatted were downloaded as well. CommonCrawl is a non-profit organisation that crawls the web, collects publicly available information and provides dataset and archives for free. A total of 664 documents were retrieved using the VirusTotal [21] service under the academic license. The files mined from the CommonCrawl set were tested with the latest versions of Avast and Kaspersky antivirus to ensure there were no malicious documents. Moreover, Rudd et al. [49] tested, their one million documents set from CommonCrawl against the VirusTotal service only found fifteen documents (0.0015 per cent) labelled as malicious. Thus, the documents collected using CommonCrawl can be primarily construed as benign.

Using the data collected the four sets were created as mentioned in Table 4.3, where the all the malicious and macro containing or macro-enabled documents (*.docm* and *.dotm* files) were considered and the percentage of *.docx* and *.dotx* documents were varied. As discussed earlier, most malware propagation in documents is still dependent on macros by maintaining an almost an

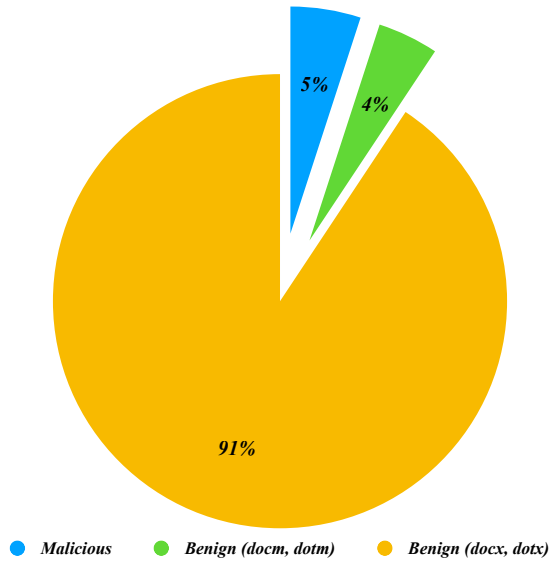


Figure 4.1: A pie chart showing the documents collected from various categories.

equal ratio of documents that may contain macros the classification bias towards such documents is reduced. Figure 4.1 provides a breakdown of all the data sampled for dataset - IV. All four of these sets were used in the standard evaluation of the detection only, for temporal evaluation documents were sampled randomly from the entire collected set to form a set of 2337 documents of which 664 were malicious. These documents were divided into training, and testing sets based on their *timeSinceCreation* field. Any document that was created on or before January 1, 2018, was part of the training set and all the documents after the selected date belonged to the testing set. Sampling and division were performed in this manner to simulate a more realistic scenario and test if the detector can predict malicious files it has yet to analyse.

4.2.2 Preprocessing

As mentioned in Section 1, each document was first validated using the Open XML SDK. This section was written in C#, which integrates with the parser, which uses a python base. It uses system calls, and standard python APIs to expand the OOXML document and XML ElementTree APIs in python to read the various XML files. This section used, oletools library [32]

script to extract the embedded macros. Features with categorical values like *Application, Template, AppVersion* and others were one-hot-encoded before classification. Along with this, while preprocessing the integrity of the file is confirmed using various timestamps from both system and file. The matplotlib library was used to create histograms for format agnostic features and entropy was calculated using Shannon's entropy (Equation 3.1).

4.2.3 Feature Extraction

Most of the features collected by the proposed method have a sizeable textual component in them. Consider the forensic features, tags like *Creator, Company, Application*, the values extracted represent significant information but would be more gainful if quantised and counted. As discussed in [58], counting occurrences of each feature like rsid-count, image-count, oleobject-count, may lead to higher count compared to other features. The timestamp features stored in the core.xml file are converted into time-since-creation, time-since-last-modified, time-since-last-printed and time-spent in the document, all measured in seconds. This variance might lead to a higher degree of freedom for adversarial attackers, and considering such a case, each of the features was standardised as mentioned below.

Data related to format agnostic features, structural features and forensic features are extracted and stored to CSV files, for other modules to process further. The extraction program was created with multiprocessing and multi-threading in mind. The time and performance vary depending on the size of the file and varies from milliseconds to up to a few minutes.

Keeping with the consistency of *Set IV* a total of 5,161 features were extracted for each file in all the sets mentioned in Table 4.3. These include 68 forensic identifiers, 289 format agnostic features and 4,804 structural features. Using feature selection techniques mentioned in Section 3.2.3, 10, 50, 75 ,100, 150, 200, 400 and 800 most prominent features were selected. These features were standardised by removing the mean and scaling to unit variance using the StandardScalar implementation in Scikit-Learn.

4.2.4 Classification

Scikit-Learn is an accessible machine learning suite, which features standard functions for classification and normalisation. These functions were used as is without any modifications. A 10 fold cross-validation was performed to rule out possible biases. That is, 90/10 split was performed on the dataset, with 90% of the documents from each set contributed towards the training while the rest were used for testing.

The tests were performed on the following four classifiers: Support Vector Machines (SVM) with linear kernel, Support Vector Machines with Radial Basis Function (RBF) kernel, Random Forests and eXtreme Gradient Boosted (XGB) trees. Historically these four classifiers have performed with great accuracy in tasks of malware classification. Training the machine learning algorithm against four different classifiers with eight different sets of features selected using three different techniques resulted in ninety-six unique testing pairs for each Dataset. For each of the classifiers, the hyper-parameters are optimised using a five-fold cross-validated grid search for optimal performance.

4.2.5 Standard Evaluation with All Feature Classes

The experiment is set up to evaluate the performance of the classifier algorithm in a lab scenario. The experiments use a randomised slicing of malicious and benign sets and thus training on all known malware.

4.2.5.1 Experiment Description

The experiment uses the dataset division mentioned in section 4.2.1 were used. The dataset is loaded into the memory using Pandas python library, where the dataset is verified and preprocessing steps like One-Hot-Encoding (OHE) and feature scaling are performed as required. The features are selected according to one of the three strategies mentioned in Section 3.2.3. Each machine learning algorithm uses certain global parameters, such as the number of trees in Random Forests. These parameters are selected using grid-search. This method is described in Section 4.2.5.2.

Table 4.4: List of parameters chosen by Grid-Search algorithm and used in the classification process

Classifier Name	Parameter Manipulated	Possible Values
Linear SVM	Penalty parameter (C)	[0.001, 0.01, 0.1, 1, 10, 100]
RBF SVM	Penalty parameter (C)	[0.001, 0.01, 0.1, 1, 10, 100]
Random Forest	Number of trees ($n\_estimators$)	[10, 100, 200, 400, 800]
XGB	Number of trees ($n\_estimators$)	[10, 100, 200, 400, 800]

4.2.5.2 Parameter Settings

The dataset was used to find the optimal parameters for each of the machine learning algorithms using five-fold cross-validated Grid-Search. Table 4.4 provides a summary of parameters chosen with grid search and all the possible values for each parameter. The set of possible values were for Random Forests, and XGB were selected using Random Search, and the number was reduced to five to improve performance. It should be noted that only one parameter for each classifier was manipulated using a grid search, and the rest of the parameters used the default values as in scikit-learn. Additionally, the *gamma* value for RBF-SVM were selected using the function i.e. $\frac{1}{n\_estimators * X.var()}$ where X is the input training set and $n\_estimators$ are several features in the set.

4.2.5.3 Results and Discussion

Table 4.5: Detection accuracy (%), False Positive Rate (FPR) and Area Under Curve (AUC) of malware using different feature selection techniques on all four generated datasets with eight different feature lengths.

DS[a]	FS[b]	FL [c]	Linear-SVM			RBF-SVM			Random Forests			XGB		
			ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC
		10	90.50	0.026	0.9175	93.08	0.048	0.9741	97.05	0.050	0.9940	96.99	0.039	0.9949
		50	91.65	0.027	0.9322	95.79	0.020	0.9886	98.37	0.033	0.9982	98.62	0.020	0.9986
		75	93.87	0.026	0.9511	96.21	0.018	0.9873	98.68	0.027	0.9979	98.68	0.015	0.9983
	CS	100	93.45	0.026	0.9515	96.09	0.018	0.9814	98.43	0.032	0.9980	98.92	0.017	0.9983

Table 4.5: Detection accuracy (%), False Positive Rate (FPR) and Area Under Curve (AUC) of malware using different feature selection techniques on all four generated datasets with eight different feature lengths. (Contd.)

DS[a]	FS[b]	FL [c]	Linear-SVM			RBF-SVM			Random Forests			XGB		
			ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC
Set I	IG	150	95.97	0.023	0.9706	96.94	0.015	0.9852	98.38	0.033	0.9984	98.74	0.018	0.9983
		200	96.15	0.020	0.9738	96.75	0.021	0.9872	98.68	0.027	0.9983	98.56	0.021	0.9991
		400	97.95	0.030	0.9923	98.08	0.029	0.9982	98.62	0.026	0.9985	99.04	0.015	0.9991
		800	98.13	0.027	0.9937	98.37	0.032	0.9975	98.56	0.027	0.9984	99.16	0.014	0.9991
		10	92.73	0.062	0.9793	97.11	0.030	0.9930	98.92	0.021	0.9981	98.92	0.017	0.9985
		50	97.59	0.017	0.9870	98.37	0.024	0.9975	98.92	0.023	0.9987	98.98	0.014	0.9992
		75	97.96	0.015	0.9880	98.38	0.024	0.9979	98.86	0.023	0.9983	98.98	0.014	0.9993
		100	97.54	0.024	0.9949	98.32	0.026	0.9974	99.10	0.020	0.9987	99.10	0.014	0.9993
		150	97.72	0.024	0.9940	98.26	0.030	0.9973	98.56	0.021	0.9973	99.04	0.015	0.9989
		200	97.72	0.020	0.9898	98.62	0.024	0.9977	98.92	0.018	0.9992	99.10	0.014	0.9993
		400	98.02	0.018	0.9918	98.37	0.029	0.9975	98.85	0.020	0.9991	99.10	0.014	0.9994
		800	97.83	0.032	0.9923	98.44	0.026	0.9971	98.85	0.018	0.9992	99.10	0.014	0.9995
	FI	10	95.90	0.083	0.9858	96.21	0.077	0.9814	97.89	0.024	0.9966	98.02	0.023	0.9973
		50	96.75	0.029	0.9843	98.07	0.023	0.9974	98.73	0.021	0.9991	99.16	0.014	0.9991
		75	97.53	0.021	0.9870	98.26	0.023	0.9970	98.73	0.021	0.9991	99.16	0.014	0.9992
		100	97.17	0.026	0.9857	98.55	0.023	0.9974	98.91	0.018	0.9991	99.10	0.015	0.9992
		150	97.29	0.024	0.9895	98.68	0.021	0.9980	98.79	0.023	0.9988	99.10	0.015	0.9992
		200	97.96	0.020	0.9911	98.67	0.020	0.9981	98.67	0.021	0.9988	99.16	0.015	0.9991
		400	98.50	0.016	0.9949	98.86	0.023	0.9980	98.61	0.024	0.9985	99.16	0.015	0.9989
		800	98.49	0.015	0.9950	98.74	0.024	0.9977	98.74	0.029	0.9987	99.16	0.015	0.9991
	CS	10	88.85	0.032	0.9094	93.77	0.065	0.9811	97.92	0.057	0.9928	97.97	0.038	0.9957
		50	90.33	0.051	0.9343	95.75	0.038	0.9904	98.15	0.048	0.9978	98.46	0.026	0.9988
		75	93.00	0.032	0.9484	95.93	0.041	0.9911	98.28	0.045	0.9975	98.73	0.023	0.9990
		100	92.87	0.039	0.9502	96.16	0.032	0.9880	97.79	0.048	0.9936	98.55	0.027	0.9989
		150	95.75	0.032	0.9682	96.66	0.023	0.9895	98.01	0.047	0.9927	98.55	0.026	0.9987
		200	95.76	0.020	0.9766	97.06	0.023	0.9893	98.46	0.045	0.9976	99.01	0.021	0.9990

Table 4.5: Detection accuracy (%), False Positive Rate (FPR) and Area Under Curve (AUC) of malware using different feature selection techniques on all four generated datasets with eight different feature lengths. (Contd.)

DS[a]	FS[b]	FL [c]	Linear-SVM			RBF-SVM			Random Forests			XGB		
			ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC
Set II	IG	400	98.19	0.023	0.9913	98.78	0.023	0.9982	98.51	0.035	0.9985	99.23	0.017	0.9993
		800	98.24	0.026	0.9942	98.83	0.032	0.9975	98.55	0.035	0.9985	99.23	0.017	0.9994
		10	96.39	0.092	0.9859	98.10	0.020	0.9964	99.19	0.017	0.9987	99.32	0.015	0.9994
		50	98.24	0.023	0.9921	99.05	0.023	0.9977	99.19	0.021	0.9994	99.19	0.015	0.9990
		75	98.69	0.021	0.9946	98.69	0.023	0.9974	99.23	0.021	0.9994	99.23	0.015	0.9993
		100	98.78	0.020	0.9945	98.64	0.027	0.9978	99.32	0.020	0.9995	99.28	0.014	0.9991
		150	98.15	0.027	0.9938	98.73	0.029	0.9978	99.28	0.020	0.9993	99.37	0.014	0.9992
		200	98.46	0.023	0.9934	98.60	0.030	0.9974	99.05	0.018	0.9992	99.32	0.015	0.9989
		400	98.28	0.039	0.9936	98.74	0.030	0.9976	98.92	0.023	0.9994	99.32	0.015	0.9989
		800	98.33	0.039	0.9942	98.55	0.029	0.9970	99.14	0.018	0.9992	99.32	0.015	0.9991
	FI	10	96.38	0.096	0.9890	96.84	0.092	0.9786	98.51	0.026	0.9973	98.42	0.026	0.9938
		50	97.20	0.033	0.9897	98.33	0.026	0.9964	99.01	0.021	0.9994	99.32	0.011	0.9985
		75	97.88	0.030	0.9873	98.55	0.027	0.9974	98.42	0.029	0.9971	99.28	0.014	0.9989
		100	98.06	0.030	0.9902	98.60	0.027	0.9979	99.05	0.021	0.9988	99.23	0.017	0.9992
		150	97.74	0.041	0.9917	98.83	0.024	0.9985	98.73	0.029	0.9987	99.32	0.014	0.9989
		200	98.15	0.026	0.9922	98.92	0.021	0.9986	98.24	0.033	0.9964	99.32	0.015	0.9990
		400	98.64	0.030	0.9957	98.96	0.023	0.9984	98.78	0.029	0.9987	99.28	0.015	0.9993
		800	98.51	0.018	0.9958	98.74	0.033	0.9979	98.60	0.036	0.9991	99.32	0.015	0.9991
	CS	10	88.38	0.050	0.9106	94.13	0.099	0.9849	98.58	0.060	0.9926	98.74	0.042	0.9960
		50	92.11	0.087	0.9488	96.63	0.053	0.9923	98.80	0.053	0.9970	99.04	0.030	0.9985
		75	93.11	0.066	0.9548	96.78	0.056	0.9919	98.86	0.045	0.9933	98.95	0.032	0.9988
		100	93.92	0.065	0.9597	96.48	0.057	0.9901	98.71	0.057	0.9974	98.74	0.041	0.9990
		150	96.60	0.039	0.9773	97.17	0.036	0.9905	98.52	0.053	0.9949	99.01	0.030	0.9987
		200	96.87	0.048	0.9721	97.56	0.044	0.9909	98.80	0.059	0.9986	99.25	0.029	0.9983
		400	98.46	0.039	0.9906	98.94	0.029	0.9977	98.85	0.042	0.9982	99.37	0.021	0.9994
		800	98.52	0.032	0.9926	98.95	0.032	0.9962	98.98	0.038	0.9986	99.40	0.021	0.9995

Table 4.5: Detection accuracy (%), False Positive Rate (FPR) and Area Under Curve (AUC) of malware using different feature selection techniques on all four generated datasets with eight different feature lengths. (Contd.)

DS[a]	FS[b]	FL [c]	Linear-SVM			RBF-SVM			Random Forests			XGB		
			ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC
Set III	IG	10	97.44	0.084	0.9920	98.61	0.045	0.9975	99.49	0.015	0.9998	99.55	0.014	0.9993
		50	98.40	0.038	0.9928	98.82	0.024	0.9982	99.10	0.024	0.9979	99.52	0.015	0.9991
		75	98.76	0.020	0.9942	99.37	0.024	0.9987	99.34	0.024	0.9972	99.58	0.014	0.9995
		150	98.68	0.047	0.9955	99.37	0.024	0.9980	99.40	0.026	0.9993	99.52	0.014	0.9991
		100	98.79	0.021	0.9936	99.34	0.023	0.9982	99.49	0.023	0.9990	99.49	0.012	0.9993
		200	98.73	0.036	0.9945	99.31	0.027	0.9973	99.28	0.021	0.9990	99.58	0.014	0.9992
		400	98.43	0.044	0.9920	99.13	0.030	0.9975	99.28	0.020	0.9973	99.52	0.014	0.9992
		800	98.46	0.041	0.9921	99.07	0.027	0.9967	99.25	0.024	0.9990	99.55	0.014	0.9992
	FI	10	97.32	0.090	0.9912	97.74	0.093	0.9867	99.31	0.017	0.9987	99.49	0.017	0.9985
		50	97.50	0.083	0.9911	98.62	0.039	0.9959	99.22	0.021	0.9994	99.43	0.020	0.9990
		75	98.28	0.033	0.9897	98.79	0.036	0.9973	98.97	0.027	0.9960	99.43	0.020	0.9992
		100	98.25	0.039	0.9899	98.91	0.032	0.9978	99.13	0.026	0.9961	99.46	0.020	0.9994
		150	98.61	0.036	0.9912	98.94	0.036	0.9978	99.01	0.035	0.9989	99.43	0.018	0.9993
		200	98.40	0.036	0.9911	99.10	0.032	0.9980	98.97	0.036	0.9980	99.43	0.018	0.9994
		400	98.89	0.041	0.9969	98.91	0.033	0.9979	99.01	0.035	0.9989	99.49	0.018	0.9994
		800	98.76	0.044	0.9949	98.95	0.045	0.9976	98.92	0.033	0.9965	99.49	0.017	0.9994
	CS	10	95.00	0.998	0.8020	98.58	0.152	0.9798	99.52	0.077	0.9825	99.66	0.053	0.9965
		50	95.35	0.904	0.9411	98.98	0.105	0.9907	99.65	0.071	0.9949	99.68	0.048	0.9992
		75	95.64	0.791	0.9492	99.16	0.089	0.9915	99.63	0.071	0.9953	99.71	0.047	0.9987
		100	97.70	0.306	0.9525	99.05	0.113	0.9904	99.59	0.080	0.9971	99.71	0.045	0.9988
		150	98.31	0.135	0.9775	98.99	0.123	0.9896	99.58	0.081	0.9899	99.73	0.044	0.9991
		200	99.32	0.090	0.9885	99.49	0.075	0.9936	99.52	0.053	0.9986	99.75	0.030	0.9993
		400	99.44	0.068	0.9928	99.65	0.044	0.9975	99.22	0.047	0.9967	99.79	0.026	0.9995
		800	99.50	0.059	0.9970	99.61	0.047	0.9961	99.48	0.045	0.9976	99.77	0.024	0.9998
		10	99.43	0.068	0.9961	99.61	0.042	0.9993	99.68	0.020	0.9984	99.83	0.024	0.9991
		50	99.42	0.069	0.9919	99.68	0.036	0.9992	99.67	0.023	0.9981	99.83	0.023	0.9994

Table 4.5: Detection accuracy (%), False Positive Rate (FPR) and Area Under Curve (AUC) of malware using different feature selection techniques on all four generated datasets with eight different feature lengths. (Contd.)

DS[a]	FS[b]	FL [c]	Linear-SVM			RBF-SVM			Random Forests			XGB		
			ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC
Set IV	IG	75	99.47	0.069	0.9961	99.78	0.027	0.9994	99.65	0.027	0.9981	99.82	0.023	0.9995
		100	99.56	0.056	0.9975	99.74	0.032	0.9990	99.65	0.026	0.9981	99.82	0.018	0.9995
		150	99.52	0.053	0.9966	99.78	0.030	0.9979	99.67	0.026	0.9981	99.80	0.018	0.9995
		200	99.56	0.054	0.9972	99.80	0.029	0.9974	99.48	0.023	0.9993	99.81	0.017	0.9994
		400	99.62	0.041	0.9949	99.80	0.030	0.9972	99.61	0.027	0.9980	99.81	0.018	0.9994
		800	99.53	0.053	0.9971	99.74	0.044	0.9973	99.58	0.032	0.9981	99.81	0.018	0.9994
	FI	10	99.34	0.089	0.9857	99.27	0.140	0.9901	99.60	0.033	0.9957	99.65	0.032	0.9973
		50	99.37	0.081	0.9920	99.46	0.060	0.9980	99.56	0.035	0.9993	99.77	0.024	0.9992
		75	99.37	0.081	0.9909	99.59	0.044	0.9981	99.04	0.038	0.9991	99.77	0.024	0.9993
		100	99.37	0.081	0.9913	99.59	0.039	0.9977	99.04	0.038	0.9982	99.76	0.024	0.9995
		150	99.36	0.083	0.9923	99.64	0.035	0.9980	99.02	0.041	0.9912	99.74	0.020	0.9994
		200	99.37	0.083	0.9919	99.62	0.035	0.9976	99.04	0.041	0.9975	99.80	0.021	0.9996
		400	99.62	0.044	0.9951	99.70	0.033	0.9970	99.04	0.039	0.9990	99.82	0.021	0.9997
		800	99.65	0.045	0.9975	99.67	0.044	0.9968	99.51	0.045	0.9981	99.78	0.021	0.9997

[a] Selection of Dataset from Table 4.3

[b] Feature Selection Technique from Section 3.2.3

[c] Feature Length

Each classifier was individually evaluated with the features selected based on the score assigned by the Chi-Square(CS), Information Gain (IG) and Feature Importance (FI) feature selection techniques. The performance was measured for each classifier with eight different feature lengths (10, 50, 75, 100, 150, 200, 400, 800) and four different sets of documents. Hyperparameters were selected using GridSearch as described above, and stratified 10-fold cross-validation was performed, i.e. the ratio of classes in each fold remains constant in each fold. The results of this experiment are produced in Table 4.5. Additionally, it should be noted, the values of accuracy, False Positive

Rate, Area Under the Curve, Precision, Recall and F-Score were calculated by averaging over scores calculated in each fold of the cross-validation process.

It was observed Dataset-I when tested produced the highest accuracy of 99.16% with 0.014 FPR and 0.9992 AUC for FL = 75 features on FI selection technique with XGB classifier as shown in Table4.5. Further, it should be noted that with XGB classifier in the same set as above, the accuracy of 99.16% was also attained with 800 features using CS feature selection. It should be noted, with these parameters, the computation overload for 800 features is high at the same time there is a high probability of over-fitting. As anticipated, all four classifiers performed the worst when given ten features. CS feature selection technique though matching FI with the highest accuracy performed worse in all other cases. The accuracy for ranged from 91.65% to 99.16%, 97.59% to 99.10%, and 96.57% to 99.16% for CS, IG and FI respectively for feature lengths 50 and above. Additionally, Linear SVM was unable to detect with acceptable accuracy for Set-I across all feature selection techniques and performed worse when paired up with CS.

With Dataset-II, the highest accuracy of 99.37% with 0.014 FPR, 0.997 TPR, and 0.9992 AUC for 150 features on IG feature selection with XGB classifier. Just like in the previous dataset, the CS performs the worst in feature selection while FI provided with some comparable results with XGB classifier. It should be noted in the same test scenario, with FL = 50 and FI feature selection with XGB classifier resulted in the least false positive rate (FPR = 0.011) of the set with a reduction in accuracy of 0.05%. Linear and RBF SVM implementations performed adequately well but suffered from relatively low accuracy, AUC and high false positive rates. Random Forest classifier performed almost at the same level as compared to XGB in specific configurations but suffered from a higher FPR.

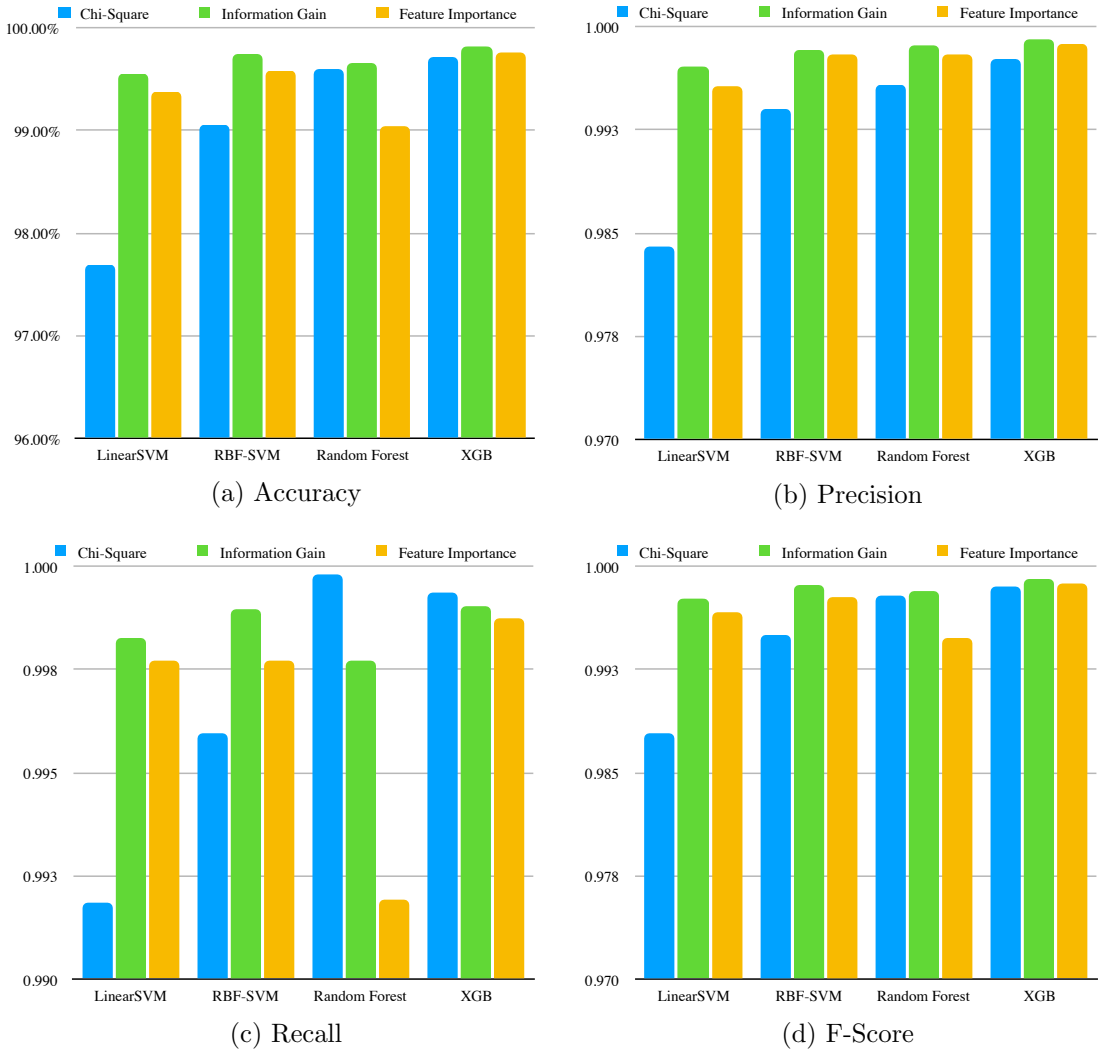
With Dataset-III, the highest accuracy of 99.58% was achieved with 0.014 FPR and 0.9995 AUC for 75 features on IG feature selection with XGB classifier. Similar results were also found with 200 features but with a reduced AUC of 0.9992. Random Forest classifier performed comparably well and had the highest accuracy of 99.45% for 100 and ten features with IG feature selection. SVM with RBF kernel performed adequately and achieved the highest accuracy of 99.37% for 150

features selected using IG. The linear method performed the worst in this set, with the highest accuracy of 98.89% (FPR = 0.044 AUC 0.9969) with FI feature selection and 400 features.

In the implementation of ALDOCX [46], authors mention how in a realistic deployment scenario, the malicious to benign ratio is skewed towards benign documents heavily. In their experiments, the dataset was populated by ∽98% benign documents and only ∽2% malicious documents. Dataset-IV represents this scenario. With the XGB classifier and 100 features selected through IG, the maximum accuracy (99.82%) with the least FPR (0.018) is achieved. It should be noted that this is a biased set, and if someone were to pick a document at random, they would have a 95% chance of picking a benign document. However, each percentage point above 95% is challenging, and for most configurations, the accuracy has been above 99% with 28 values being more than 99.7% accurate. The least FPR (0.017) is attained by XGB with IG and 200 features, while the maximum AUC is achieved in the same configuration but with 400 features.

Throughout all configurations, these patterns have been observed. Except for configurations where Linear-SVM and feature length is 10, all the configurations have an AUC higher than 0.99. The Chi-Square based feature selection underperforms except in the case of high values of FL when paired up with XGB classifier. Additionally, the Linear SVM based classifiers perform the worst across the board, thus pointing towards the utility of non-linear classifiers. With the increase in benign samples in each of the dataset, the accuracy also increases but at the same time, FPR rate and AUC value remain constant. This indicates the classifier learning patterns and not 'memorising' or over-fitting.

The malware recognition proficiency of different classifiers measured with different evaluation metrics, such as Recall, Precision, and F-Score, is shown in Table 4.6. Recall presents the rate of identification of positive instances correctly by the classifiers and ranges from 0 to 1, higher being the better. While, precision measures the relevance of the detection, by computing a ratio of correctly identified documents with the total number of documents identified correctly belonging to a class. F-Score is calculated by taking the harmonic mean of precision and recall and is used to measure a configuration's accuracy for a classifier. All three of these metrics take a value between



(a) Accuracy (b) Precision

(c) Recall (d) F-Score

Figure 4.2: Malware detection evaluation using different performance metrics for FL = 100 on the Dataset-IV using all three feature sets.

0 and 1 with higher value signifying better results. The expanded table which shows results for all the configurations is available in Table A.1.

Incidentally, the XGB classifier produced the highest accuracy for Dataset-IV with IG feature selection in this configuration. The IG selection technique performed exceptionally well while maximising for accuracy, Precision and F-score, and Recall when combined with the SVM based classifier. The CS performed well on the Recall metric when combined with Tree-based methods. It should be noted that all the Recall values under consideration here are higher than 0.990, which

is in itself a great score. The XGB classifier outperformed other classifiers with the accuracy of 99.82% with near perfect Precision of 0.999, high Recall of 0.9990, and F-score value equal to 0.999. The minimum performance was shown by LinearSVM when paired up with CS selection approach, where the worst accuracy with an accuracy of 97.70%, Precision of 0.984, Recall of 0.9918 and F-Score of 0.988.

Table 4.6: Experiment results of malware detection with different feature selection techniques for Dataset-IV with all three feature sets

FS[a]	FL [b]	Linear-SVM			RBF-SVM			Random Forests			XGB		
		Prec [c]	Recall	F-1 [d]	Prec [c]	Recall	F-1 [d]	Prec [c]	Recall	F-1 [d]	Prec [c]	Recall	F-1[d]
CS	75	0.960	0.9956	0.978	0.995	0.9958	0.996	0.996	0.9998	0.998	0.998	0.9994	0.998
	100	0.984	0.9918	0.988	0.994	0.9960	0.995	0.996	0.9998	0.998	0.998	0.9994	0.998
	150	0.993	0.9893	0.991	0.994	0.9959	0.995	0.996	0.9998	0.998	0.998	0.9994	0.999
	200	0.995	0.9976	0.996	0.996	0.9986	0.997	0.997	0.9977	0.997	0.998	0.9990	0.999
IG	75	0.996	0.9980	0.997	0.999	0.9991	0.999	0.999	0.9977	0.998	0.999	0.9993	0.999
	150	0.997	0.9977	0.997	0.998	0.9993	0.999	0.999	0.9979	0.998	0.999	0.9989	0.999
	100	0.997	0.9983	0.998	0.998	0.9990	0.999	0.999	0.9977	0.998	0.999	0.9990	0.999
	200	0.997	0.9982	0.998	0.998	0.9994	0.999	0.999	0.9957	0.997	0.999	0.9989	0.999
FI	75	0.996	0.9976	0.997	0.998	0.9980	0.998	0.998	0.9919	0.995	0.999	0.9988	0.999
	100	0.996	0.9977	0.997	0.998	0.9977	0.998	0.998	0.9919	0.995	0.999	0.9987	0.999
	150	0.996	0.9976	0.997	0.998	0.9980	0.998	0.998	0.9918	0.995	0.999	0.9983	0.999
	200	0.996	0.9977	0.997	0.998	0.9978	0.998	0.998	0.9921	0.995	0.999	0.9990	0.999

[a] Feature Selection Technique from Section 3.2.3

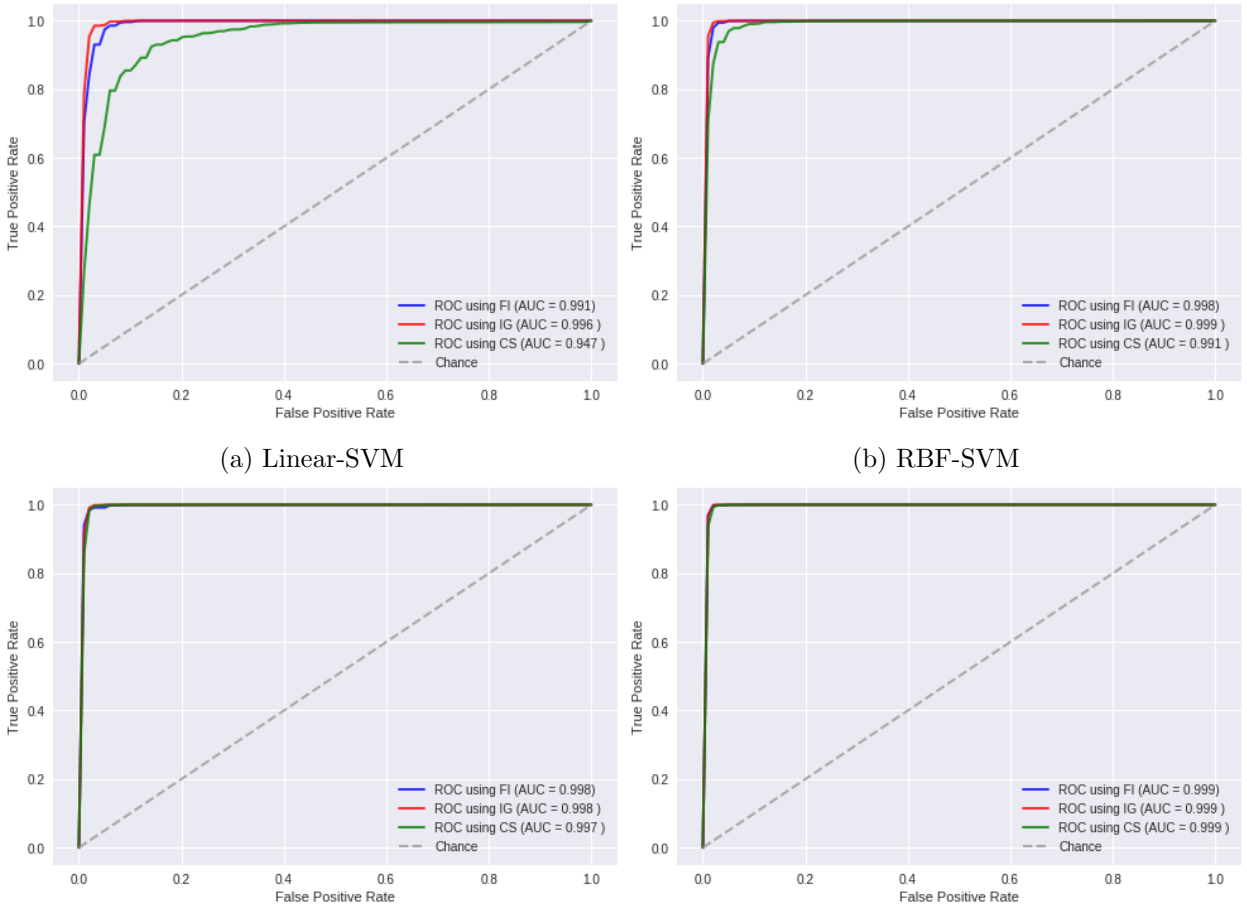
[b] Feature Length

[c] Precision

[d] F-1 Score

The classifiers' performance was evaluated separately for each feature selection technique over

the above-mentioned performance metrics as well as accuracy using 100 features on Dataset-IV. The results of this evaluation are shown in Figure 4.2.



(a) Linear-SVM

(b) RBF-SVM

(c) Random Forest

(d) XGB

Figure 4.3: ROC curves for feature selection techniques used by different classifiers for 100 features on Datasets-IV.

The Receiver Operating Characteristic (ROC) curve depicts a trade-off TPR and FPR of a particular configuration. Performance of each classifier was evaluated by plotting the ROC curves with Dataset-IV and 100 feature length configuration, and these plots are shown in Figure 4.3. The Area Under the Curve (AUC) metric, calculated for all possible configurations in Table 4.5, measures the ROC curve. In one of the unfavourable scenarios, AUC is 0.5, which is equivalent to picking a class at random, whereas high values of AUC represent better accuracy for each classifier. Ideally, a classifier which attains high values of TPR at shallow levels of FPR is considered excellent.

Figure 4.3 presents the ROC curve of individual classifiers such as Linear SVM, RBF-SVM, Random Forest, and XGB for Dataset-IV and 100 features. Each ROC curve of the classifier corresponds to a separate feature selection technique.

4.2.5.4 Conclusion

As discussed earlier, the proposed method performed with excellent accuracy. With Dataset - IV it reported accuracy of 99.82% which far exceeds the current state of the art methodology. In all configurations, AUC of upwards of 0.99 was reported. All these facts support our Hypothesis - 1, declared in Section 3.1, the feature representation presented is robust and can sufficiently detect the presence of malicious documents.

4.2.6 Standard Evaluation with Forenisc Identifiers

4.2.6.1 Experiment Description

Similar to Section 4.2.5, each of the three non-linear classifiers were individually evaluated with the features selected based on the score assigned by the Chi-Square(CS), Information Gain (IG) and Feature Importance (FI) feature selection techniques for just the forensic features as mentioned in Table 3.4.

As expressed in Section 4.2.5, the Linear SVM performed the worse in all three parameters when compared to the other non-linear classifiers; for this reason, only non-linear classifiers were chosen. The performance was measured for each classifier with six different feature lengths (10, 15, 20, 35, 50, 64) and four different sets of documents. Hyperparameters were selected using Grid-Search as described above, and stratified 10-fold cross-validation was performed, i.e. the ratio of classes in each fold remains constant in each fold. It should be noted that features in consideration are a subset of all the features previously discussed and used in Section 4.2.5. The results of this experiment are produced in Table 4.7. Additionally, it should be noted, the values of accuracy, False Positive Rate, Area Under the Curve, Precision, Recall and F-Score were calculated by averaging over scores calculated in each fold of the cross-validation process.

4.2.6.2 Result and Discussion

In Dataset-I and just 50 forensic features selected through FI the accuracy of 99.28% with 0.014 FPR and 0.9981 AUC was achieved using Random Forest classifiers. When compared to the results in the dataset I of Table 4.5, it may be noted there is an increase of 0.12% accuracy as well as a decrease of 0.0011 in AUC which is very minute. RBF-SVM reported the highest accuracy (98.92%) when 50 features were selected using FI selection while the highest AUC and lowest FPR for 35 features selected using FI selection. XGB achieved the highest accuracy of 99.10% with 20 features using IG selection. With Dataset- II, with XGB classifier accuracy of 99.32% was attained four times, with FPR 0.012 and 0.9989 AUC. When all 68 of the forensic identifiers were used with any feature selection technique, the accuracy of 99.10% was achieved.

Additionally, 33 of the 54 values in Table 4.7 have an accuracy higher or equivalent to 99.10%. When compared to the accuracy values attained in the Dataset-II using all three feature sets, there is a minor difference in accuracy. With using all three feature sets, the highest accuracy of 99.37% was achieved using 150 features, i.e. there was a gain of 0.05% in comparison to just using forensic features. Additionally, it should also be noted that only 23 Dataset-II configurations in Table 4.5 had an accuracy higher than 99.10%.

In Dataset-III, XGB classifier outperformed the others, achieving an accuracy of 99.55% with 0.015 FPR and 0.9986 AUC using only ten features using IG feature selection technique. Compared to its counterpart in the Table 4.5 the accuracy is identical, while the FPR and AUC values are more favourable. As it is expected when compared to the highest value achieved using all three sets, Section 4.2.5 outperforms in every category with close margins. With the Dataset-IV, the highest accuracy of 99.83% was achieved using both 15 features from CS and FI selection methods and XGB classifier. The CS method reported an FPR of 0.029 and AUC of 0.9990, whereas FI method reported an FPR of 0.020 and AUC of 0.9988.

Table 4.7: Detection accuracy (%), False Positive Rate (FPR) and Area Under Curve (AUC) of malware using different feature selection techniques on all four generated datasets with six different feature lengths for just the forensic features

DS[a]	FS[b]	FL [c]	RBF-SVM			Random Forests			XGB		
			ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC
Set I	CS	10	96.57	0.039	0.9922	98.98	0.014	0.9987	98.74	0.015	0.9986
		15	97.29	0.023	0.9933	99.04	0.017	0.9985	98.80	0.014	0.9988
		20	97.30	0.023	0.9879	99.10	0.015	0.9973	98.80	0.014	0.9988
		35	98.80	0.009	0.9988	99.16	0.015	0.9987	98.92	0.015	0.9991
		50	98.80	0.014	0.9987	98.98	0.015	0.9958	98.86	0.015	0.9988
		68	98.74	0.017	0.9973	99.22	0.015	0.9986	98.86	0.015	0.9990
	IG	10	95.91	0.038	0.9889	98.92	0.017	0.9975	98.92	0.015	0.9995
		15	96.27	0.033	0.9896	98.86	0.020	0.9987	98.80	0.015	0.9992
		20	97.42	0.014	0.9897	98.74	0.021	0.9974	99.10	0.014	0.9993
		35	98.14	0.017	0.9963	99.10	0.020	0.9987	99.04	0.014	0.9986
		50	98.32	0.026	0.9966	99.10	0.020	0.9986	98.92	0.015	0.9986
		68	98.74	0.017	0.9973	99.10	0.018	0.9987	98.86	0.015	0.9990
	FI	10	97.83	0.023	0.9945	99.10	0.015	0.9974	98.80	0.014	0.9971
		15	97.66	0.023	0.9936	99.16	0.015	0.9986	99.04	0.012	0.9983
		20	98.08	0.021	0.9943	99.16	0.015	0.9989	98.98	0.015	0.9984
		35	98.80	0.010	0.9993	99.16	0.015	0.9986	98.98	0.014	0.9990
		50	98.92	0.015	0.9973	99.28	0.014	0.9981	98.86	0.015	0.9990
		68	98.74	0.017	0.9973	99.22	0.015	0.9987	98.86	0.015	0.9990
	CS	10	96.52	0.050	0.9923	98.96	0.018	0.9981	98.83	0.018	0.9985
		15	96.93	0.047	0.9918	99.14	0.021	0.9988	98.87	0.021	0.9989
		20	97.16	0.041	0.9907	99.32	0.020	0.9987	99.01	0.017	0.9987

Table 4.7: Detection accuracy (%), False Positive Rate (FPR) and Area Under Curve (AUC) of malware using different feature selection techniques on all four generated datasets with six different feature lengths for just the forensic features (Contd.)

DS[a]	FS[b]	FL [c]	RBF-SVM			Random Forests			XGB		
			ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC
Set II		35	98.74	0.010	0.9984	99.28	0.021	0.9988	99.14	0.017	0.9987
		50	98.92	0.012	0.9971	99.28	0.018	0.9982	99.10	0.017	0.9986
		68	98.78	0.023	0.9979	99.10	0.020	0.9959	99.32	0.012	0.9989
	IG	10	97.88	0.020	0.9958	99.23	0.020	0.9983	99.14	0.014	0.9988
		15	98.69	0.021	0.9980	99.28	0.020	0.9989	99.19	0.015	0.9991
		20	98.55	0.023	0.9979	99.19	0.021	0.9988	99.19	0.015	0.9992
		35	98.65	0.017	0.9985	99.23	0.023	0.9979	99.28	0.014	0.9989
		50	98.92	0.021	0.9973	99.19	0.023	0.9986	99.19	0.014	0.9990
		68	98.78	0.023	0.9979	99.23	0.021	0.9989	99.32	0.012	0.9989
	FI	10	97.20	0.042	0.9926	99.19	0.020	0.9990	99.05	0.021	0.9984
		15	98.06	0.023	0.9959	99.23	0.020	0.9982	99.14	0.017	0.9985
		20	98.19	0.033	0.9951	99.28	0.018	0.9989	99.10	0.015	0.9984
		35	99.10	0.017	0.9981	99.28	0.021	0.9993	99.23	0.015	0.9988
		50	98.87	0.020	0.9979	99.14	0.018	0.9962	99.32	0.012	0.9989
		68	98.78	0.023	0.9979	99.23	0.021	0.9989	99.32	0.012	0.9989
	CS	10	97.65	0.051	0.9941	99.40	0.020	0.9979	99.10	0.023	0.9990
		15	98.04	0.053	0.9938	99.40	0.023	0.9983	99.37	0.023	0.9983
		20	98.07	0.050	0.9936	99.43	0.024	0.9982	99.31	0.023	0.9984
		35	98.61	0.047	0.9986	99.43	0.023	0.9989	99.46	0.015	0.9988
		50	99.28	0.026	0.9989	99.49	0.023	0.9990	99.49	0.017	0.9991
		68	99.22	0.026	0.9989	99.43	0.024	0.9982	99.49	0.017	0.9989

Table 4.7: Detection accuracy (%), False Positive Rate (FPR) and Area Under Curve (AUC) of malware using different feature selection techniques on all four generated datasets with six different feature lengths for just the forensic features (Contd.)

DS[a]	FS[b]	FL [c]	RBF-SVM			Random Forests			XGB		
			ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC
Set III	IG	10	98.37	0.065	0.9967	99.40	0.023	0.9991	99.55	0.015	0.9986
		15	98.52	0.050	0.9979	99.43	0.023	0.9983	99.46	0.018	0.9987
		20	98.61	0.051	0.9983	99.37	0.024	0.9992	99.43	0.020	0.9985
		35	98.98	0.029	0.9985	99.43	0.024	0.9992	99.49	0.017	0.9988
		50	99.01	0.030	0.9979	99.37	0.023	0.9949	99.49	0.017	0.9989
		68	99.22	0.026	0.9989	99.46	0.020	0.9943	99.49	0.017	0.9989
	FI	10	97.77	0.098	0.9932	99.46	0.018	0.9984	99.49	0.017	0.9988
		15	98.37	0.072	0.9974	99.43	0.023	0.9993	99.49	0.017	0.9986
		20	98.52	0.057	0.9976	99.40	0.023	0.9965	99.43	0.017	0.9987
		35	99.31	0.020	0.9989	99.43	0.024	0.9989	99.49	0.015	0.9989
		50	99.31	0.023	0.9990	99.49	0.023	0.9978	99.49	0.017	0.9989
		68	99.22	0.026	0.9989	99.49	0.023	0.9991	99.49	0.017	0.9989
Set IV	CS	10	98.98	0.099	0.9917	99.80	0.036	0.9980	99.75	0.036	0.9988
		15	99.05	0.098	0.9931	99.82	0.035	0.9964	99.83	0.029	0.9990
		20	99.16	0.093	0.9948	99.11	0.024	0.9949	99.78	0.020	0.9985
		35	99.50	0.074	0.9971	99.13	0.026	0.9971	99.74	0.020	0.9990
		50	99.62	0.042	0.9986	99.51	0.027	0.9993	99.71	0.021	0.9992
		68	99.67	0.038	0.9991	99.13	0.026	0.9986	99.73	0.020	0.9994
	IG	10	99.20	0.069	0.9975	99.12	0.023	0.9980	99.79	0.020	0.9993
		15	99.32	0.069	0.9980	99.13	0.024	0.9951	99.77	0.020	0.9992
		20	99.43	0.053	0.9978	99.10	0.027	0.9987	99.76	0.021	0.9990

Table 4.7: Detection accuracy (%), False Positive Rate (FPR) and Area Under Curve (AUC) of malware using different feature selection techniques on all four generated datasets with six different feature lengths for just the forensic features (Contd.)

DS[a]	FS[b]	FL [c]	RBF-SVM			Random Forests			XGB		
			ACC	FPR	AUC	ACC	FPR	AUC	ACC	FPR	AUC
		35	99.56	0.044	0.9980	99.54	0.027	0.9973	99.75	0.021	0.9991
		50	99.53	0.042	0.9979	99.10	0.029	0.9987	99.76	0.021	0.9993
		68	99.67	0.038	0.9991	99.65	0.026	0.9995	99.73	0.020	0.9994
	FI	10	99.11	0.092	0.9946	99.13	0.018	0.9988	99.80	0.021	0.9985
		15	99.10	0.092	0.9941	99.13	0.024	0.9980	99.83	0.020	0.9988
		20	99.36	0.092	0.9966	99.11	0.024	0.9986	99.77	0.020	0.9986
		35	99.67	0.038	0.9983	99.73	0.026	0.9981	99.72	0.018	0.9993
		50	99.71	0.033	0.9992	99.18	0.024	0.9952	99.73	0.020	0.9994
		68	99.67	0.038	0.9991	99.65	0.026	0.9995	99.73	0.020	0.9994

[a] Selection of Dataset from Table 4.3

[b] Feature Selection Technique from Section 3.2.3

[c] Feature Length

Malware recognition proficiency of all three classifiers measured metrics such as Precision, Recall and F-Score is reported in Table 4.8. As previously mentioned, Precision measures the relevance of the detection, recall measures the rate of identification for positive instances marked correctly by the classifier and F-Score the accuracy of the configuration by taking the harmonic mean of Precision and Recall.

The classifiers' performance was evaluated separately for each feature selection technique over the above-mentioned performance metrics as well as accuracy using 15 features on Dataset-IV. The results of this evaluation are shown in Figure 4.5. Incidentally, the XGB classifier produced the highest accuracy for Dataset-IV with FI feature selection in this configuration. Unlike in Section

Table 4.8: Experiment results of malware detection with different feature selection techniques for Dataset-IV with only forensic features

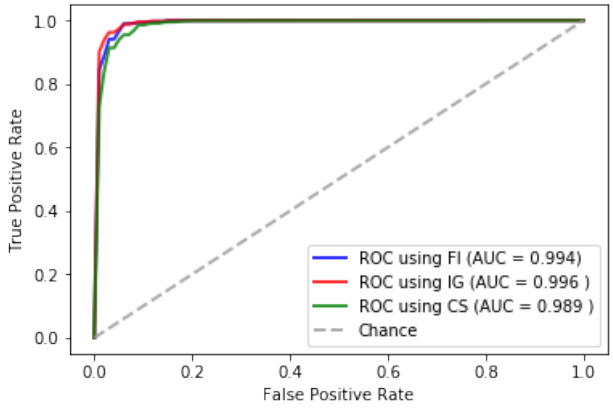
FS[a]	FL [b]	RBF-SVM			Random Forests			XGB		
		Prec [c]	Recall	F-1 [d]	Prec [c]	Recall	F-1 [d]	Prec	Recall	F-1[d]
CS	10	0.995	0.9945	0.995	0.998	0.9998	0.999	0.998	0.9993	0.999
	15	0.995	0.9952	0.995	0.998	0.9999	0.999	0.998	0.9997	0.999
	20	0.995	0.9961	0.996	0.999	0.9919	0.995	0.999	0.9987	0.999
	35	0.996	0.9987	0.997	0.999	0.9921	0.995	0.999	0.9983	0.999
	50	0.998	0.9983	0.998	0.999	0.9963	0.997	0.999	0.9981	0.998
IG	10	0.996	0.9952	0.996	0.999	0.9919	0.995	0.999	0.9988	0.999
	15	0.996	0.9965	0.996	0.999	0.9921	0.995	0.999	0.9987	0.999
	20	0.997	0.9967	0.997	0.999	0.9920	0.995	0.999	0.9986	0.999
	35	0.998	0.9976	0.998	0.999	0.9966	0.998	0.999	0.9985	0.999
	50	0.998	0.9972	0.997	0.998	0.9920	0.995	0.999	0.9986	0.999
FI	10	0.995	0.9955	0.995	0.999	0.9918	0.995	0.999	0.9990	0.999
	15	0.995	0.9954	0.995	0.999	0.9921	0.995	0.999	0.9993	0.999
	20	0.995	0.9981	0.997	0.999	0.9919	0.995	0.999	0.9987	0.999
	35	0.998	0.9985	0.998	0.999	0.9985	0.999	0.999	0.9980	0.999
	50	0.998	0.9987	0.998	0.999	0.9926	0.996	0.999	0.9982	0.999

[a] Feature Selection Technique from Section 3.2.3
[b] Feature Length
[c] Precision
[d] F-1 Score

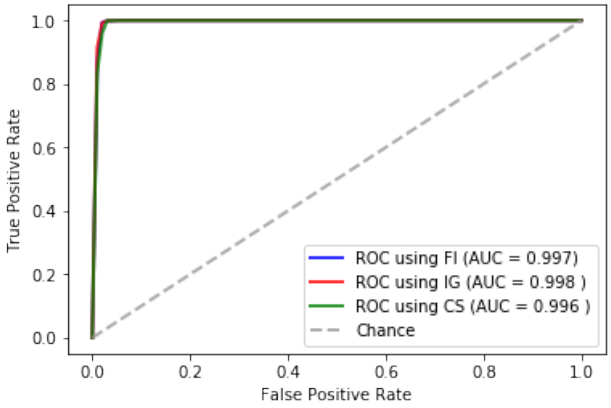
4.2.5, all three feature selection techniques performed well. When paired up the XGB classifier, both CS and FI techniques perform well with an accuracy of 99.82%; similar results are obtained by pairing Random Forests with CS. When optimising for Precision, both IG and FI perform similarly when paired with tree-based methods, while IG outperforms the other two when paired with RBF-SVM. CS either outperformed or matched the performance of FI, against Recall and F-Score metrics when XGB was used. Random Forests paired with CS selection matched the performance XGB paired with CS or FI. XGB paired with FI reported accuracy of 99.83% with near perfect Precision of 0.999, high Recall of 0.9993, and F-score value equal to 0.999. The minimum performance was shown by RBF-SVM when paired up with CS selection approach, where the accuracy was 99.05%, Precision was 0.995, Recall was 0.9952 and F-Score was 0.995.

The Receiver Operating Characteristic (ROC) curve depicts a trade-off TPR and FPR of a

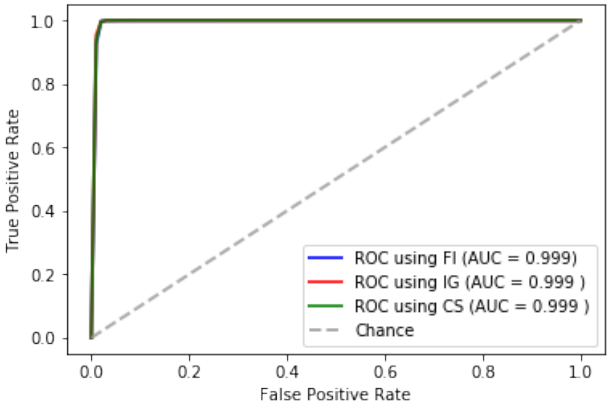
particular configuration. Performance of each classifier was evaluated by plotting the ROC curves with Dataset-IV and 15 feature length configuration; and these plots are shown in Figure 4.4. The AUC metric, calculated for all possible configurations in Table 4.7, quantifies the ROC curve tradeoff. In one of the unfavourable scenarios, AUC is 0.5, which is equivalent to picking a class at random, whereas high values of AUC represent better accuracy for each classifier. Ideally, a classifier which attains high values of TPR at deficient levels of FPR is considered excellent. Figure 4.3 presents the ROC curve of individual classifiers such as RBF-SVM, Random Forest, and XGB for Dataset-IV and 15 features. Each ROC curve of the classifier corresponds to a separate feature selection technique.



(a) RBF-SVM



(b) Random Forest



(c) XGB

Figure 4.4: ROC curves for feature selection techniques used by different classifiers for 15 features on Datasets-IV using only forensic features.

From the above discussion and Tables 4.5 and 4.7, these observations can be made. Unlike Section 4.2.5 CS feature selection method is more effective in the case of just forensic identifiers than the combination of all the three feature sets. Even though forensic identifiers are part of Section 4.2.5, it is highly probable that all forensic identifiers are not selected. The performance results from this experiment are a testament to the quality of forensic features identified. In comparison to the *set of all features*, which was explored in Section 4.2.5, just using forensic identifiers decreases the number of features thereby increasing the computational efficiency while maintaining the performance from Section 4.2.5. It should be further noted that just using forensic features is highly inadvisable. Though effective, they may be susceptible to the adversarial approach. Most of the forensic features are extracted from XML files and in a perfect-knowledge white-box attack scenario, these features can be manipulated with little to no cost. Thus, it is highly recommended to use at least one more feature representation.

4.2.6.3 Conclusion

As discussed earlier, the proposed method performed well in detecting malicious documents. With just the forensic identifiers and Dataset - IV, a consistent accuracy of over 99% was maintained throughout. All these facts support our hypothesis - H1, declared in Section 3.1, just metadata and forensic identifiers are sufficient to identify malicious documents.

4.3 Temporal Analysis of the Proposed Method

The proposed method was trained with the dataset of files that were created before a specific date and was tested against the files generated after this date. This evaluation tests the generalisability of the system that is if the system can predict and detect future novel attacks based on the current level of information.

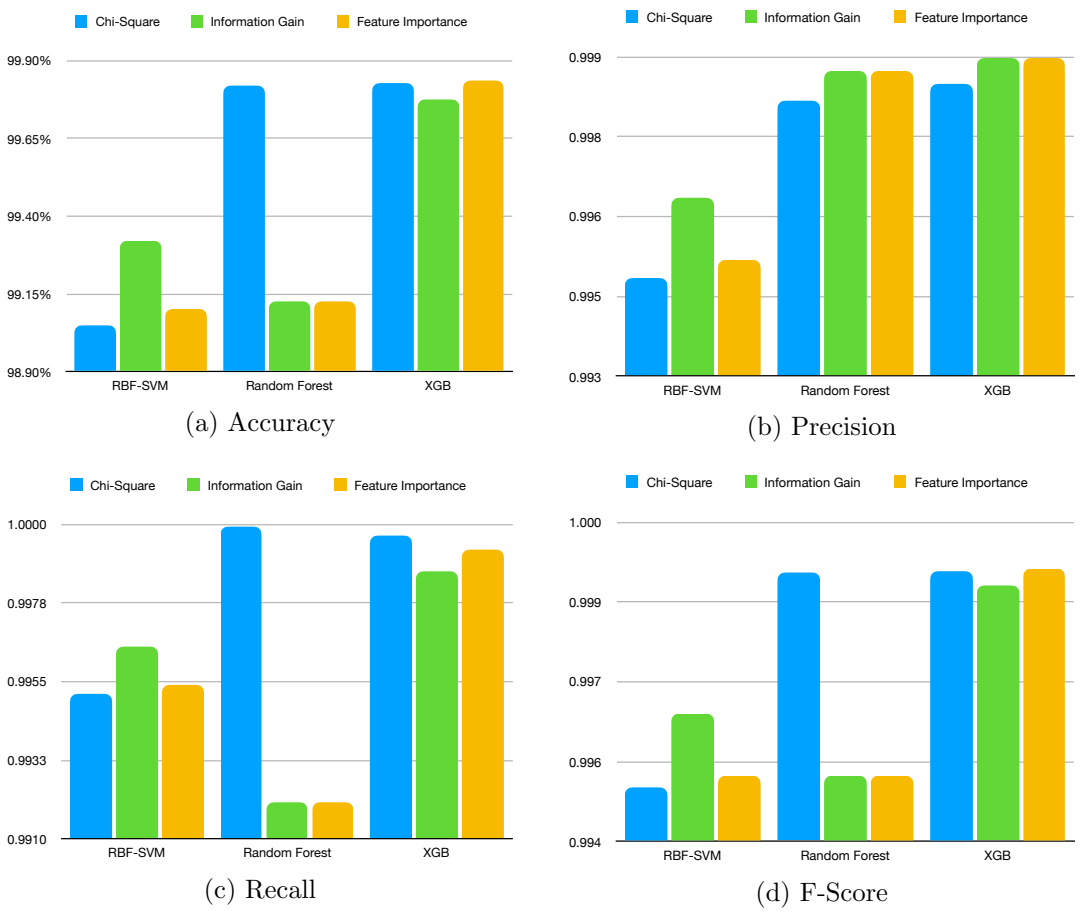


Figure 4.5: Malware detection evaluation using different performance metrics for FL = 15 on the Dataset-IV just using forensic identifiers.

4.3.1 Experiment Description

Temporal evaluation is used to predict the ability of the model to predict attacks that previously have not been seen by the detection system. These attacks include attacks created using zero-day vulnerabilities and other novel approaches. Additionally, as [64] discusses, in most realistic deployments, all possible malware configurations are rarely known, and thus generalisability of the system should also be tested. To this end, malicious documents set is split on the basis of *timeSinceCreation* field present in the documents as it could easily split both malicious and benign sets effectively. Documents that were created before January 1, 2018, were added to the training set (630 samples) while the documents created after were added to the testing set (34 samples). 1673

documents were sampled from the benign set and were added to each set, leading to 1886 training samples and 451 samples. As seen through testing in standard evaluation, the Information Gain and Feature Importance selection methods produced better results. As previously mentioned section 4.2.5.2, the hyper-parameters are calculated using five-fold cross-validation, while the training and testing are performed without any cross-validation.

4.3.2 Result and Discussion

The goal of the analysis was to assess the predictive performance of the proposed method against unknown attacks rather than general performance. Table 4.9 shows the results of this analysis. As it is quite evident, when the results of this evaluation are compared to those from Dataset-II from Table 4.5, there has been a decrement of accuracy from $\sim 3\%$ to up to 6.2%. The stagnation of performance in each column with an increase in features is an indicator of over-fitting of the model. Malware authors often use an existing benign file or sample as a base and inject malicious code in these files for their attack, when these files would contain the indicators of the benign base file, while it was created much later. By selecting *timeSinceCreation* as the splitting feature over *first_seen in VirusTotal* might have lead to an imbalance in distribution and over-fitted results. It was also discovered *timeSinceCreation* and *created* tags are prone to marked incorrectly. Didriksen [2] noted documents that were converted to the OOXML format show inconsistencies when *created* tag is concerned as well as software like Microsoft Office Online suite records this date incorrectly consistently.

4.3.3 Conclusion

As stated earlier, some drop-off in accuracy is expected when shifting to the temporal analysis. In this case, the highest accuracy was recorded with 200 features at 96.45% when Random Forests and Information Gain feature selection was used. The stagnation in the learning curve was also pointed out and attributed to mislabelled metadata. All these facts partially support our hypothesis H2, declared in Section 3.1, the *created* field stored within the document can be used to segregate

Table 4.9: Detection accuracy (%) of malware classifier using all features reduced to eight feature length sets using information gain and feature importance for the temporal dataset

Dataset	Feature Length	Information Gain		Feature Importance	
		RF-Accuracy	XGB-Accuracy	RF-Accuracy	XGB-Accuracy
Temporal	10	94.68	94.24	93.79	92.24
Temporal	50	93.79	94.01	95.12	94.01
Temporal	75	93.79	94.01	94.24	95.57
Temporal	100	94.46	94.68	94.68	95.12
Temporal	150	96.01	94.46	94.68	94.90
Temporal	200	**96.45**	94.46	94.46	96.01
Temporal	400	96.23	94.46	94.46	94.46
Temporal	800	95.12	94.46	94.68	96.01

the documents temporally, but the internal inconstancies of the field suggest an alternate should be sought after.

4.4 Overall Conclusion

In Chapter 2, four research gaps were identified. Based on these, a new method of detection was developed, which depends on domain-specific knowledge of the OOXML document format. Section 4.1 clearly shows that the current generation of anti-virus software is incapable of detecting malicious documents. The research gaps and hypothesis were tested in Sections 4.2.5 and 4.2.6.

Research gap R1 identifies the lack of detector which uses domain understanding to extract features from the malicious documents, while R2 suggests the lack of customisability at the time of deployment. These gaps were extended to hypothesis H1, where the use of domain-specific features like metadata and forensic identifiers was suggested. Sections 4.2.5 and 4.2.6 prove these features can be used to detect malicious documents and at the same time, they can be used to customise if need be. Research gap R3 identifies the lack of temporal analysis and generalizability testing in previous research. Hypothesis H2 was formed where the use of metadata tags such as *created* *lastPrinted* and *modified* was suggested. Using the *created* tag present in A temporal analysis of detection technique was performed, and it resulted in minimal decreased performance. Research gap R4 identifies the lack of data, both malicious and benign, while training and testing malicious

documents in the past. A large sampling of data was collected through malware services as well as mining the internet. It is thereby concluded the research gaps R1, R2 and R4 presented in Chapter 2 were resolved while gap R3 was partially resolved.

Chapter 5

Discussions and Conclusion

A static detection method for OOXML file format by extracting domain specific features was presented. The system uses metadata and forensic identifiers with some of the previously analysed state-of-the-art features (i.e., byte-histograms, byte-entropy histograms and some structural features). The system is independent of external dependencies like external parsers. Four sets of experiments were performed on the proposed method. It can be concluded that the proposed method performs more accurately than current antivirus solutions available. In the standard evaluation, using forensic features and all three feature classes, the maximum accuracy of 99.8% was recorded, which is better than the current state of the art methodology available in research. In the temporal evaluation, the generalisability of the system using a temporal dataset was tested where there was a decrease in accuracy of about 3% in the best case.

5.1 Research Outcome

During the course of this thesis, two research articles were communicated to international journals. The titles of these articles are as follows:

(1) Singh P. and Tapaswi S., "Detection of Malicious Office Documents Employing Forensic Identifiers" (Communicated)

(2) Singh P., Tapaswi S., and Gupta, S., "Malware Detection in PDF and Office Documents: A Survey" (Communicated)

5.2 Contributions and Discussion

The proposed system used static malware analysis on Microsoft office documents, to identify malicious documents with high accuracy. A large corpus of benign documents was collected using CommonCrawl APIs, and malicious documents were collected from the VirusTotal service. The use of forensic identifiers in OOXML documents in the detection of malicious documents OOXML was introduced with this work. As discussed earlier in Section 4.2.6, just forensic identifiers performed exceptionally well across all four sets of documents. With the highest accuracy (99.83%) and AUC (0.9988) reached for fifteen features and XGB classifier. With XGB classifier and just forensic features on Dataset-IV the value of accuracy varied between 99.72 % and 99.83%, while the FPR ranged between 0.036 and 0.018. The Area under the ROC curve had a value higher than 0.99 for each possible configuration in Dataset-IV. As noted earlier in Table 4.8, most configurations have high precision, recall and F-score as well as proving the utility of the classifier.

The salient advantage of using these features, in particular, would be the ease of feature extraction, as most of these features are stored directly as XML files with UTF-8 format and performance benefit of using a maximum of 68 features. It should be of note, the list of forensic identifiers is inexhaustive, and the 68 features suggested in this work are just reference and numerous other features of forensic note can be added in the representation. These advantages present a double-edged sword, as malware authors can also with the same ease modify these identifiers to their advantage. Thus, as it was suggested earlier, more robust methods are required, which may stand against adversarial approaches or make it computationally very expensive for the attacker to evade. The robustness may be achieved by cascading classifiers or using more resilient feature representation.

As presented in Section 4.2.5, the combination byte-entropy-histograms, byte histograms and structural features along with forensic identifiers represent the document through multiple dimensions, thus making it harder to evade. As previously discussed, they perform either better or at the same level as just forensic features. As it can be ascertained from Table 4.5, XGB performs

the best in the group. With respect to Dataset-IV, the accuracy varies between 99.66% and 99.83%, while the lowest FPR of 0.017 was achieved with 200 features and the corresponding accuracy of 99.81%. Table 4.6 summarised the exceptional values of Precision, Recall and F-Score, and in most configurations, there is a high yield.

Table 4.9 summarises the temporal analysis of the proposed method across numerous features. As it was noted in the Section 4.3, there was a decrease in accuracy from $\sim$ 3% to up to 6.2% when compared to it is nearest counterpart (Dataset-II) in the standard analysis. This decrease is expected and has often been seen in research [58]. Additionally, the Table 4.9, displayed signs of overfitting, it is hypothesised the over-fitting occurred due to the use of a substandard splitting feature.

5.2.1 Comparison to Existing Work

In their work [46], Nissim et al. the authors created a corpus of 16,811 documents out of which only 327 documents are malicious (1.94%). They used Structural Feature Extraction Methodology [47] for feature extraction. They extracted 134,854 unique features and used information gain for feature selection. It has been pointed by Miura et al. [48], ALDOCX would fail if the corpus considered had both malicious and benign macros. Further, they also pointed out that the methodology is also susceptible to mislabelling malicious documents that masquerading as benign documents. With 100 features selected using SVM classifier, they reported accuracy of 99.67% FPR of 0.0019 and TPR of 0.9334. F-Score was calculated using the reported results and found to be 0.9569. When compared with the best results in Table 4.5 and 4.6, the following observations can be made. The proposed method outperformed ALDOCX when compared on accuracy and TPR, while ALDOCX reported a finer FPR. The proposed method reported an F-Score of 0.9995, thereby making it a better method of the two.

There are no direct metrics on which there can be a comparison with [49], but the following observations were made. As MEADE has only used format agnostic based features, malicious documents that implement downloaders may be missed. These type of malware do not have the

whole malicious code embedded with them, but instead, use some vulnerability or social engineering to transfer control of the program so malware can be downloaded. DDE scripts are one of many ways to create this form of attack. Additionally, the methodology presented in [49] calculates multiple entropies, histograms and hashes and thus would be computationally intensive. Both the method presented by them as well as the proposed method reported an AUC of greater than 0.99, albeit in different configurations.

5.3 Limitations and Future scope

Although this work provides excellent results when tested over various Microsoft Word formatted documents, there are some limitations which can provide future research directions. The features discussed in this work are universal for all popular office file extensions but only tested them against various Microsoft Word documents. Performance against different file formats in the office suite can be calculated in the future.

Adversarial learning is often used to create adversarial samples with the aim to subvert the detector in operation. This operation is achieved by changing the feature representation of malware, so it closely resembles that of the benign class, so the classifier skews towards misclassifying the file. The adversary is often used to retrain the detector model to be more resilient against these forms of attacks. The features introduced in this work have not been tested against an adversarial network, and thus their robustness is unknown.

The proposed methodology is highly flexible and allows for the addition of numerous other identifiers that can be added to improve the performance and robustness further. The list of forensic and metadata identifiers presented in this work are limited and scratches the surface of the possible XML tags presented in ECMA-376, which can be appropriated into detection systems. Based on the needs of the implementation, more forensic and metadata identifiers can be researched and augmented into the framework.

Bibliography

[1] J. Paoli, I. Valet-Harper, A. Farquhar, and I. Sebestyen, "Ecma-376 office open xml file formats," URL http://www. ecmainternational. org/publications/standards/Ecma-376. htm, p. 11, 2006.

[2] E. Didriksen, "Forensic analysis of ooxml documents," Master's thesis, Gjøvik University College, 2014.

[3] D. Rentz. (2007, 08) Microsoft compound document file format. [Online]. Available: https://www.openoffice.org/sc/compdocfileformat.pdf

[4] A. Harper, S. Harris, J. Ness, C. Eagle, G. Lenkey, and T. Williams, Gray hat hacking: the ethical hacker's handbook. McGraw-Hill Education, 2018.

[5] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in IKT 2013 - 5th Conference on Information and Knowledge Technology, 05 2013, pp. 113–120.

[6] R. Wash and M. M. Cooper, "Who provides phishing training?: Facts, stories, and people like me," in Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. ACM, 2018, p. 492.

[7] TrustWave. (2018, 11) 2018 trustwave global security report. Accessed: 04 Feburary 2019. [Online]. Available: https://www2.trustwave.com/GlobalSecurityReport.html

[8] Verizon. (2018, 03) 2018 data breach investigations report. Accessed: 04 Feburary 2019. [Online]. Available: https://goo.gl/QgWSBn

[9] Sophos. (2018, 04) Sophoslabs 2019 threat report. Accessed: 04 Feburary 2019. [Online]. Available: https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophoslabs-2019-threat-report.pdf

[10] Symantec. (2007, 02) W97m.melissa.a. Accessed: 2018-08-15. [Online]. Available: https://www.symantec.com/security-center/writeup/2000-122113-1425-99

[11] J. Dechaux, E. Filiol, and J.-P. Fizaine, "Office documents: New weapons of cyberwarfare," 2010.

[12] K. Zetter. (2016, 03) Inside the cunning, unprecedented hack of ukraine's power grid. [Online]. Available: https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/

[13] K. J. Higgins. (2016, 01) Macros, network sniffers, but still no 'smoking gun' in ukraine blackout. [Online]. Available: https://www.darkreading.com/threat-intelligence/macros-network-sniffers-but-still-no-smoking-gun-in-ukraine-blackout/d/d-id/1324076

[14] McAfee. (2018, 06) Mcafee labs threats report. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-jun-2018.pdf

[15] Windows-Defender-Research. (2016, 03) New feature in office 2016 can block macros and help prevent infection. [Online]. Available: https://cloudblogs.microsoft.com/microsoftsecure/2016/03/22/new-feature-in-office-2016-can-block-macros-and-help-prevent-infection/

[16] Cisco. (2018, 02) Cisco 2018 annual cyber security report. Accessed: 2018-09-19. [Online]. Available: http://www.cisco.com/go/acr2018

[17] C. "Cimpanu. "kaspersky: 70 percent of attacks now target office vulnerabilities". Accessed: 15 April 2019. [Online]. Available: https://www.zdnet.com/article/kaspersky-70-percent-of-attacks-now-target-office-vulnerabilities/

[18] K. Kuczma. Microsoft targeted by 8 of 10 top vulnerabilities in 2018. Accessed: 15 April 2019. [Online]. Available: https://go.recordedfuture.com/hubfs/reports/cta2019-0319.pdf

[19] Sophos. (2017, 11) Sophoslabs 2018 malware forecast. [Online]. Available: https://www.sophos.com/en-us/en-us/medialibrary/PDFs/technical-papers/malware-forecast-2018.pdf

[20] R. Sherstobitoff. (2018, 02) Lazarus resurfaces, targets global banks and bitcoin users. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-jun-2018.pdf

[21] VirusTotal. Accessed: 27 November 2019. [Online]. Available: https://www.virustotal.com/

[22] CommonCrwal. Accessed: November 2018. [Online]. Available: http://commoncrawl.org

[23] E. Brumaghin. (2017, 10) Spoofed sec emails distribute evolved dnsmessenger. Accessed: May 2019. [Online]. Available: https://blog.talosintelligence.com/2017/10/dnsmessenger-sec-campaign.html

[24] G. Szappanos. (2015, 08) Microsoft word intruder revealed. [Online]. Available: www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophos-microsoft-word-intruder-revealed.pdf

[25] J. H. Nart Villeneuve. (2015, 01) A new word document exploit kit. [Online]. Available: https://www.fireeye.com/blog/threat-research/2015/04/a_new_word_document.html

[26] G. Szappanos. . akbuilder–the crowdsourced exploit kit. Accessed: 15 April 2019. [Online]. Available: https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/AKBuilder-public.pdf

[27] G. "Szappanos. "akbuilder – the crowdsourced exploit kit". Accessed: 15 April 2019. [Online]. Available: https://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/sophos-office-exploit-generators-szappanos.pdf

[28] C. Cimpanu. Kaspersky: 70 percent of attacks now target office vulnerabilities. Accessed: 21 April 2019. [Online]. Available: https://www.zdnet.com/article/is-it-still-a-good-idea-to-publish-proof-of-concept-code-for-zero-days/

[29] I. A. Saeed, A. Selamat, and A. M. Abuagoub, "A survey on malware and malware detection systems," International Journal of Computer Applications, vol. 67, no. 16, 2013.

[30] L. Zeltser. (2017, 09) Malware analysis cheat sheet. [Online]. Available: https://digital-forensics.sans.org/media/analyzing-malicious-document-files.pdf

[31] Y. Prayudi, I. Riadi et al., "Implementation of malware analysis using static and dynamic analysis method," International Journal of Computer Applications, vol. 117, no. 6, 2015.

[32] P. "Lagadec. "oletools - python tools to analyze ole and ms office files". Accessed: 27 January 2019. [Online]. Available: https://www.decalage.info/python/oletools

[33] Microsoft-Corporation. (2018, 08) [ms-doc]: Word (.doc) binary file format. [Online]. Available: https://interoperability.blob.core.windows.net/files/MS-DOC/[MS-DOC].pdf

[34] Microsoft. (2007, 10) 2007 office document: Open xml markup explained. [Online]. Available: https://www.microsoft.com/en-za/download/details.aspx?id=15359

[35] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016, pp. 785–794.

[36] T. Chen, T. He, M. Benesty, V. Khotilovich, and Y. Tang, "Xgboost: extreme gradient boosting," R package version 0.4-2, pp. 1–4, 2015.

[37] R. Bearden and D. C.-T. Lo, "Automated microsoft office macro malware detection using machine learning," in Big Data (Big Data), 2017 IEEE International Conference on. IEEE, 2017, pp. 4448–4452.

[38] W.-J. Li, S. Stolfo, A. Stavrou, E. Androulaki, and A. D. Keromytis, "A study of malcode-bearing documents," in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2007, pp. 231–250.

[39] M. Z. Shafiq, S. A. Khayam, and M. Farooq, "Embedded malware detection using markov n-grams," in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2008, pp. 88–107.

[40] T. Schreck, S. Berger, and J. Göbel, "Bissam: Automatic vulnerability identification of office documents," in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2012, pp. 204–213.

[41] B. Gu, Y. Fang, P. Jia, L. Liu, L. Zhang, and M. Wang, "A new static detection method of malicious document based on wavelet package analysis," in Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2015 International Conference on. IEEE, 2015, pp. 333–336.

[42] P. Lagadec, "Opendocument and open xml security (openoffice. org and ms office 2007)," Journal in Computer Virology, vol. 4, no. 2, pp. 115–125, 2008.

[43] A. M. Naser, M. H. Btoush, and A. H. Hadi, "Analyzing and detecting malicious content: Docx files," International Journal of Computer Science and Information Security, vol. 14, no. 8, p. 404, 2016.

[44] J.-Y. Lin and H.-K. Pao, "Multi-view malicious document detection," in Technologies and Applications of Artificial Intelligence (TAAI), 2013 Conference on. IEEE, 2013, pp. 170–175.

[45] D. Maiorca, G. Giacinto, and I. Corona, "A pattern recognition system for malicious pdf files detection," in International Workshop on Machine Learning and Data Mining in Pattern Recognition. Springer, 2012, pp. 510–524.

[46] N. Nissim, A. Cohen, and Y. Elovici, "Aldocx: detection of unknown malicious microsoft office documents using designated active learning methods based on new structural feature extraction methodology," IEEE Transactions on Information Forensics and Security, vol. 12, no. 3, pp. 631–646, 2017.

[47] A. Cohen, N. Nissim, L. Rokach, and Y. Elovici, "Sfem: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods," Expert Systems with Applications, vol. 63, pp. 324–343, 2016.

[48] H. Miura, M. Mimura, and H. Tanaka, "Macros finder: Do you remember loveletter?" in International Conference on Information Security Practice and Experience. Springer, 2018, pp. 3–18.

[49] E. M. Rudd, R. Harang, and J. Saxe, "Meade: Towards a malicious email attachment detection engine," arXiv preprint arXiv:1804.08162, 2018.

[50] S. D. l. Santos and J. Torres, "Macro malware detection using machine learning techniques - a new approach," in Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,, INSTICC. SciTePress, 2017, pp. 295–302.

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

[52] E. Aboud and D. O'Brien, "Detection of malicious vba macros using machine learning methods," 2018.

[53] K. Z. Snow, S. Krishnan, F. Monrose, and N. Provos, "Shellos: Enabling fast detection and forensic analysis of code injection attacks." in USENIX Security Symposium, 2011, pp. 183–200.

[54] K. Iwamoto and K. Wasaki, "A method for shellcode extraction from malicious document files using entropy and emulation," International Journal of Engineering and Technology, vol. 8, no. 2, p. 101, 2016.

[55] Z. Fu, X. Sun, Y. Liu, and B. Li, "Forensic investigation of ooxml format documents," Digital Investigation, vol. 8, no. 1, pp. 48–55, 2011.

[56] Oracle. (2015, 05) Zero day malware threat prevention ensuring document safety with outside in clean content. [Online]. Available: http://www.oracle.com/us/products/middleware/zero-day-malware-protection-brief-2607983.pdf

[57] B. Biggio, I. Corona, B. Nelson, B. I. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, and F. Roli, "Security evaluation of support vector machines in adversarial environments," in Support Vector Machines Applications. Springer, 2014, pp. 105–153.

[58] D. Maiorca, B. Biggio, M. E. Chiappe, and G. Giacinto, "Adversarial detection of flash malware: Limitations and open issues," arXiv preprint arXiv:1710.10225, 2017.

[59] H. S. Anderson and P. Roth, "Ember: An open dataset for training static pe malware machine learning models," arXiv preprint arXiv:1804.04637, 2018.

[60] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on. IEEE, 2015, pp. 11–20.

[61] Microsoft. Welcome to the open xml sdk 2.5 for office. Accessed: 13 December 2018. [Online]. Available: https://docs.microsoft.com/en-us/office/open-xml/open-xml-sdk

[62] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in Icml, vol. 97, no. 412-420, 1997, p. 35.

[63] J. R. Quinlan, "Induction of decision trees," Machine learning, vol. 1, no. 1, pp. 81–106, 1986.

[64] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "Tesseract: Eliminating experimental bias in malware classification across space and time," arXiv preprint arXiv:1807.07838, 2018.

Appendix A

Precision, Recall and F-Score Tables

A.1 Precision, Recall and F-Score Table for Section 4.2.5

Table A.1: Precision, Recall and F-Score of malware detection method with three feature selection techniques on all four generated datasets with eight different feature lengths and all three feature classes.

DS[a]	FS[b]	FL [c]	Linear-SVM			RBF-SVM			Random Forests			XGB		
			Prec [d]	Recall	F-1 [e]	Prec [d]	Recall	F-1 [e]	Prec [d]	Recall	F-1 [e]	Prec [d]	Recall	F-1[e]
SET I	CS	10	0.980	0.8588	0.914	0.966	0.9168	0.940	0.968	0.9840	0.976	0.974	0.9760	0.975
		50	0.980	0.8788	0.925	0.986	0.9429	0.964	0.978	0.9950	0.987	0.987	0.9900	0.988
		75	0.982	0.9149	0.946	0.988	0.9489	0.967	0.982	0.9960	0.989	0.990	0.9880	0.989
		100	0.982	0.9079	0.942	0.988	0.9469	0.966	0.979	0.9950	0.987	0.989	0.9930	0.991
		150	0.985	0.9479	0.965	0.990	0.9590	0.974	0.978	0.9950	0.987	0.988	0.9910	0.989
		200	0.986	0.9489	0.967	0.986	0.9599	0.972	0.982	0.9960	0.989	0.986	0.9900	0.988
		400	0.980	0.9859	0.983	0.981	0.9870	0.984	0.983	0.9940	0.989	0.990	0.9940	0.992
		800	0.982	0.9870	0.984	0.979	0.9940	0.987	0.982	0.9940	0.988	0.991	0.9950	0.993
	IG	10	0.958	0.9199	0.937	0.980	0.9719	0.976	0.986	0.9960	0.991	0.989	0.9930	0.991
		50	0.989	0.9709	0.980	0.984	0.9890	0.986	0.985	0.9970	0.991	0.991	0.9920	0.991
		75	0.990	0.9760	0.983	0.984	0.9890	0.986	0.985	0.9960	0.991	0.991	0.9920	0.991
		100	0.984	0.9749	0.979	0.983	0.9890	0.986	0.987	0.9980	0.993	0.991	0.9940	0.992
		150	0.984	0.9780	0.981	0.980	0.9910	0.986	0.986	0.9900	0.988	0.990	0.9940	0.992
		200	0.987	0.9750	0.981	0.984	0.9930	0.989	0.988	0.9940	0.991	0.991	0.9940	0.992
		400	0.988	0.9790	0.983	0.981	0.9920	0.987	0.987	0.9940	0.990	0.991	0.9940	0.992

		800	0.979	0.9850	0.982	0.983	0.9910	0.987	0.988	0.9929	0.990	0.991	0.9940	0.992
	FI	10	0.948	0.9869	0.967	0.951	0.9880	0.969	0.984	0.9809	0.982	0.985	0.9820	0.983
		50	0.981	0.9649	0.973	0.985	0.9830	0.984	0.986	0.9930	0.989	0.991	0.9950	0.993
		75	0.986	0.9729	0.979	0.985	0.9860	0.985	0.986	0.9930	0.989	0.991	0.9950	0.993
		100	0.983	0.9699	0.976	0.985	0.9910	0.988	0.988	0.9940	0.991	0.990	0.9950	0.992
		150	0.984	0.9709	0.977	0.986	0.9920	0.989	0.985	0.9950	0.990	0.990	0.9950	0.992
		200	0.987	0.9789	0.983	0.987	0.9910	0.989	0.986	0.9919	0.989	0.990	0.9960	0.993
		400	0.989	0.9859	0.987	0.985	0.9960	0.991	0.984	0.9930	0.989	0.990	0.9960	0.993
		800	0.990	0.9849	0.987	0.984	0.9950	0.990	0.981	0.9980	0.990	0.990	0.9960	0.993
SET II	CS	10	0.984	0.8542	0.913	0.971	0.9387	0.954	0.976	0.9948	0.985	0.984	0.9871	0.986
		50	0.976	0.8838	0.927	0.983	0.9555	0.969	0.980	0.9942	0.987	0.989	0.9890	0.989
		75	0.985	0.9135	0.947	0.982	0.9593	0.971	0.981	0.9948	0.988	0.990	0.9916	0.991
		100	0.982	0.9148	0.947	0.986	0.9587	0.972	0.980	0.9890	0.984	0.988	0.9910	0.990
		150	0.986	0.9529	0.969	0.990	0.9619	0.976	0.980	0.9916	0.986	0.989	0.9903	0.990
		200	0.991	0.9477	0.969	0.990	0.9677	0.979	0.981	0.9974	0.989	0.991	0.9948	0.993
		400	0.990	0.9839	0.987	0.990	0.9923	0.991	0.985	0.9935	0.989	0.993	0.9961	0.995
		800	0.989	0.9858	0.987	0.987	0.9968	0.992	0.985	0.9942	0.990	0.993	0.9961	0.995
	IG	10	0.962	0.9877	0.974	0.992	0.9813	0.986	0.993	0.9955	0.994	0.994	0.9968	0.995
		50	0.990	0.9845	0.987	0.990	0.9961	0.993	0.991	0.9974	0.994	0.994	0.9948	0.994
		75	0.991	0.9903	0.991	0.990	0.9910	0.991	0.991	0.9981	0.995	0.994	0.9955	0.995
		100	0.992	0.9910	0.991	0.988	0.9923	0.990	0.992	0.9987	0.995	0.994	0.9955	0.995
		150	0.988	0.9852	0.987	0.988	0.9942	0.991	0.992	0.9981	0.995	0.994	0.9968	0.995
		200	0.990	0.9877	0.989	0.987	0.9929	0.990	0.992	0.9942	0.993	0.994	0.9968	0.995
		400	0.983	0.9923	0.988	0.987	0.9948	0.991	0.990	0.9942	0.992	0.994	0.9968	0.995
		800	0.983	0.9929	0.988	0.988	0.9916	0.990	0.992	0.9955	0.994	0.994	0.9968	0.995
	FI	10	0.960	0.9897	0.975	0.962	0.9942	0.978	0.989	0.9897	0.989	0.989	0.9884	0.989
		50	0.986	0.9742	0.980	0.989	0.9871	0.988	0.991	0.9948	0.993	0.996	0.9948	0.995
		75	0.987	0.9826	0.985	0.988	0.9910	0.990	0.988	0.9897	0.989	0.994	0.9955	0.995
		100	0.987	0.9852	0.986	0.988	0.9916	0.990	0.991	0.9955	0.993	0.993	0.9961	0.995
		150	0.983	0.9852	0.984	0.990	0.9935	0.992	0.988	0.9942	0.991	0.994	0.9961	0.995
		200	0.989	0.9845	0.987	0.991	0.9935	0.992	0.986	0.9890	0.987	0.994	0.9968	0.995
		400	0.987	0.9935	0.990	0.990	0.9948	0.993	0.988	0.9948	0.991	0.994	0.9961	0.995

		800	0.992	0.9864	0.989	0.986	0.9961	0.991	0.985	0.9955	0.990	0.994	0.9968	0.995
SET III	CS	10	0.986	0.8671	0.922	0.975	0.9514	0.963	0.985	0.9974	0.991	0.990	0.9947	0.992
		50	0.977	0.9232	0.949	0.987	0.9710	0.979	0.987	0.9981	0.993	0.993	0.9955	0.994
		75	0.983	0.9304	0.955	0.986	0.9737	0.980	0.989	0.9970	0.993	0.992	0.9947	0.993
		100	0.983	0.9402	0.961	0.986	0.9703	0.978	0.986	0.9981	0.992	0.990	0.9944	0.992
		150	0.990	0.9673	0.978	0.991	0.9736	0.982	0.987	0.9947	0.991	0.993	0.9951	0.994
		200	0.988	0.9729	0.980	0.989	0.9804	0.985	0.986	0.9996	0.993	0.993	0.9977	0.995
		400	0.990	0.9906	0.990	0.993	0.9940	0.993	0.990	0.9962	0.993	0.995	0.9974	0.996
		800	0.992	0.9894	0.991	0.992	0.9947	0.993	0.991	0.9966	0.994	0.995	0.9977	0.996
	IG	10	0.979	0.9891	0.984	0.989	0.9940	0.991	0.996	0.9974	0.997	0.997	0.9977	0.997
		50	0.991	0.9894	0.990	0.994	0.9913	0.993	0.994	0.9947	0.994	0.996	0.9977	0.997
		75	0.995	0.9894	0.992	0.994	0.9981	0.996	0.994	0.9977	0.996	0.997	0.9981	0.997
		150	0.988	0.9951	0.992	0.994	0.9981	0.996	0.994	0.9989	0.996	0.997	0.9974	0.997
		100	0.995	0.9902	0.992	0.994	0.9974	0.996	0.994	0.9992	0.997	0.997	0.9966	0.997
		200	0.991	0.9932	0.992	0.993	0.9981	0.996	0.995	0.9962	0.995	0.997	0.9981	0.997
		400	0.989	0.9913	0.990	0.993	0.9966	0.995	0.995	0.9958	0.995	0.997	0.9974	0.997
		800	0.990	0.9910	0.990	0.993	0.9951	0.994	0.994	0.9966	0.995	0.997	0.9977	0.997
	FI	10	0.978	0.9891	0.983	0.977	0.9951	0.986	0.996	0.9955	0.996	0.996	0.9977	0.997
		50	0.980	0.9894	0.984	0.990	0.9925	0.991	0.995	0.9955	0.995	0.995	0.9977	0.996
		75	0.992	0.9868	0.989	0.991	0.9940	0.992	0.993	0.9940	0.994	0.995	0.9977	0.996
		100	0.990	0.9879	0.989	0.992	0.9943	0.993	0.994	0.9955	0.995	0.995	0.9981	0.997
		150	0.991	0.9917	0.991	0.991	0.9959	0.993	0.991	0.9962	0.994	0.996	0.9974	0.996
		200	0.991	0.9891	0.990	0.992	0.9966	0.994	0.991	0.9962	0.994	0.996	0.9974	0.996
		400	0.990	0.9962	0.993	0.992	0.9947	0.993	0.991	0.9962	0.994	0.996	0.9981	0.997
		800	0.989	0.9955	0.992	0.989	0.9981	0.993	0.992	0.9947	0.993	0.996	0.9977	0.997
	CS	10	0.950	1.0000	0.974	0.992	0.9930	0.993	0.996	0.9990	0.997	0.997	0.9992	0.998
		50	0.955	0.9987	0.976	0.994	0.9948	0.995	0.996	1.0000	0.998	0.997	0.9992	0.998
		75	0.960	0.9956	0.978	0.995	0.9958	0.996	0.996	0.9998	0.998	0.998	0.9994	0.998
		100	0.984	0.9918	0.988	0.994	0.9960	0.995	0.996	0.9999	0.998	0.998	0.9994	0.998
		150	0.993	0.9893	0.991	0.994	0.9959	0.995	0.996	0.9998	0.998	0.998	0.9994	0.999
		200	0.995	0.9976	0.996	0.996	0.9986	0.997	0.997	0.9977	0.997	0.998	0.9990	0.999
		400	0.996	0.9977	0.997	0.998	0.9986	0.998	0.998	0.9942	0.996	0.999	0.9991	0.999

		800	0.997	0.9979	0.997	0.998	0.9983	0.998	0.998	0.9969	0.997	0.999	0.9989	0.999
	IG	10	0.996	0.9975	0.997	0.998	0.9981	0.998	0.999	0.9977	0.998	0.999	0.9995	0.999
		50	0.996	0.9975	0.997	0.998	0.9986	0.998	0.999	0.9977	0.998	0.999	0.9994	0.999
		75	0.996	0.9980	0.997	0.999	0.9991	0.999	0.999	0.9977	0.998	0.999	0.9993	0.999
		150	0.997	0.9977	0.997	0.998	0.9993	0.999	0.999	0.9979	0.998	0.999	0.9989	0.999
		100	0.997	0.9983	0.998	0.998	0.9990	0.999	0.999	0.9977	0.998	0.999	0.9990	0.999
		200	0.997	0.9982	0.998	0.998	0.9994	0.999	0.999	0.9957	0.997	0.999	0.9989	0.999
		400	0.998	0.9981	0.998	0.998	0.9994	0.999	0.999	0.9973	0.998	0.999	0.9990	0.999
		800	0.997	0.9979	0.998	0.998	0.9996	0.999	0.998	0.9972	0.998	0.999	0.9990	0.999
	FI	10	0.995	0.9977	0.997	0.993	0.9997	0.996	0.998	0.9975	0.998	0.998	0.9979	0.998
		50	0.996	0.9977	0.997	0.997	0.9975	0.997	0.998	0.9972	0.998	0.999	0.9989	0.999
		75	0.996	0.9976	0.997	0.998	0.9980	0.998	0.998	0.9919	0.995	0.999	0.9988	0.999
		100	0.996	0.9977	0.997	0.998	0.9977	0.998	0.998	0.9919	0.995	0.999	0.9987	0.999
		150	0.996	0.9976	0.997	0.998	0.9980	0.998	0.998	0.9918	0.995	0.999	0.9983	0.999
		200	0.996	0.9977	0.997	0.998	0.9978	0.998	0.998	0.9921	0.995	0.999	0.9990	0.999
		400	0.998	0.9983	0.998	0.998	0.9986	0.998	0.998	0.9919	0.995	0.999	0.9992	0.999
		800	0.998	0.9987	0.998	0.998	0.9988	0.998	0.998	0.9972	0.997	0.999	0.9988	0.999

[a] Selection of Dataset from Table 4.3

[b] Feature Selection Technique from Section 3.2.3

[c] Feature Length

[d] Precision

[e] F-Score

A.2 Precision, Recall and F-Score Table for Section 4.2.6

Table A.2: Precision, Recall and F-Score of malware detection method with three feature selection techniques on all four generated datasets with eight different feature lengths and all three feature classes.

DS[a]	FS[b]	FL [c]	RBF-SVM			Random Forests			XGB		
			Prec [d]	Recall	F-1 [e]	Prec [d]	Recall	F-1 [e]	Prec [d]	Recall	F-1[e]
SET I	CS	10	0.974	0.9690	0.971	0.991	0.9920	0.991	0.990	0.9890	0.989
		15	0.985	0.9700	0.977	0.989	0.9950	0.992	0.991	0.9890	0.990
		20	0.985	0.9700	0.977	0.990	0.9950	0.993	0.991	0.9890	0.990
		35	0.994	0.9860	0.990	0.990	0.9960	0.993	0.990	0.9920	0.991
		50	0.991	0.9890	0.990	0.990	0.9930	0.991	0.990	0.9910	0.990
		68	0.989	0.9900	0.989	0.990	0.9970	0.994	0.990	0.9910	0.990
	IG	10	0.975	0.9569	0.965	0.989	0.9930	0.991	0.990	0.9920	0.991
		15	0.978	0.9599	0.968	0.987	0.9940	0.991	0.990	0.9900	0.990
		20	0.991	0.9659	0.978	0.986	0.9930	0.990	0.991	0.9940	0.992
		35	0.989	0.9800	0.984	0.987	0.9980	0.993	0.991	0.9930	0.992
		50	0.983	0.9890	0.986	0.987	0.9980	0.993	0.990	0.9920	0.991
		68	0.989	0.9900	0.989	0.988	0.9970	0.993	0.990	0.9910	0.990
	FI	10	0.985	0.9790	0.982	0.990	0.9950	0.993	0.991	0.9890	0.990
		15	0.985	0.9760	0.980	0.990	0.9960	0.993	0.992	0.9920	0.992
		20	0.986	0.9820	0.984	0.990	0.9960	0.993	0.990	0.9930	0.991
		35	0.993	0.9870	0.990	0.990	0.9960	0.993	0.991	0.9920	0.991
		50	0.990	0.9920	0.991	0.991	0.9970	0.994	0.990	0.9910	0.990
		68	0.989	0.9900	0.989	0.990	0.9970	0.994	0.990	0.9910	0.990
CS		10	0.979	0.9716	0.975	0.992	0.9929	0.993	0.992	0.9910	0.992
		15	0.980	0.9761	0.978	0.991	0.9968	0.994	0.991	0.9929	0.992
		20	0.982	0.9768	0.980	0.992	0.9987	0.995	0.993	0.9929	0.993

Table A.2: Precision, Recall and F-Score of malware detection method with three feature selection techniques on all four generated datasets with eight different feature lengths and all three feature classes.

DS[a]	FS[b]	FL [c]	RBF-SVM			Random Forests			XGB		
			Prec [c]	Recall	F-1 [d]	Prec [c]	Recall	F-1 [d]	Prec [c]	Recall	F-1[d]
		35	0.995	0.9865	0.991	0.991	0.9987	0.995	0.993	0.9948	0.994
		50	0.995	0.9897	0.992	0.992	0.9974	0.995	0.993	0.9942	0.994
		68	0.990	0.9923	0.991	0.992	0.9955	0.994	0.995	0.9955	0.995
	IG	10	0.991	0.9781	0.985	0.992	0.9974	0.995	0.994	0.9935	0.994
		15	0.991	0.9903	0.991	0.992	0.9981	0.995	0.994	0.9948	0.994
		20	0.990	0.9890	0.990	0.991	0.9974	0.994	0.994	0.9948	0.994
		35	0.993	0.9877	0.990	0.990	0.9987	0.995	0.994	0.9955	0.995
		50	0.991	0.9935	0.992	0.990	0.9981	0.994	0.994	0.9942	0.994
		68	0.990	0.9923	0.991	0.991	0.9981	0.995	0.995	0.9955	0.995
	FI	10	0.982	0.9781	0.980	0.992	0.9968	0.994	0.991	0.9955	0.993
		15	0.990	0.9819	0.986	0.992	0.9974	0.995	0.993	0.9948	0.994
		20	0.986	0.9884	0.987	0.992	0.9974	0.995	0.994	0.9935	0.994
		35	0.993	0.9942	0.994	0.991	0.9987	0.995	0.994	0.9955	0.995
		50	0.992	0.9923	0.992	0.992	0.9955	0.994	0.995	0.9955	0.995
		68	0.990	0.9923	0.991	0.991	0.9981	0.995	0.995	0.9955	0.995
	CS	10	0.987	0.9834	0.985	0.995	0.9974	0.996	0.994	0.9944	0.994
		15	0.987	0.9887	0.988	0.994	0.9981	0.996	0.994	0.9977	0.996
		20	0.988	0.9883	0.988	0.994	0.9989	0.996	0.994	0.9970	0.996
		35	0.988	0.9943	0.991	0.994	0.9985	0.996	0.996	0.9970	0.997
		50	0.994	0.9974	0.995	0.994	0.9992	0.997	0.996	0.9977	0.997
		68	0.994	0.9966	0.995	0.994	0.9989	0.996	0.996	0.9977	0.997

Table A.2: Precision, Recall and F-Score of malware detection method with three feature selection techniques on all four generated datasets with eight different feature lengths and all three feature classes.

DS[a]	FS[b]	FL [c]	RBF-SVM			Random Forests			XGB		
			Prec [c]	Recall	F-1 [d]	Prec [c]	Recall	F-1 [d]	Prec [c]	Recall	F-1[d]
	IG	10	0.984	0.9959	0.990	0.994	0.9981	0.996	0.996	0.9981	0.997
		15	0.988	0.9940	0.991	0.994	0.9985	0.996	0.996	0.9977	0.997
		20	0.987	0.9955	0.991	0.994	0.9981	0.996	0.995	0.9977	0.996
		35	0.993	0.9943	0.994	0.994	0.9989	0.996	0.996	0.9977	0.997
		50	0.993	0.9951	0.994	0.994	0.9977	0.996	0.996	0.9977	0.997
		68	0.994	0.9966	0.995	0.995	0.9981	0.997	0.996	0.9977	0.997
	FI	10	0.976	0.9966	0.986	0.995	0.9977	0.997	0.996	0.9977	0.997
		15	0.982	0.9977	0.990	0.994	0.9985	0.996	0.996	0.9977	0.997
		20	0.986	0.9959	0.991	0.994	0.9981	0.996	0.996	0.9970	0.996
		35	0.995	0.9962	0.996	0.994	0.9989	0.996	0.996	0.9974	0.997
		50	0.994	0.9970	0.996	0.994	0.9992	0.997	0.996	0.9977	0.997
		68	0.994	0.9966	0.995	0.994	0.9992	0.997	0.996	0.9977	0.997
	CS	10	0.995	0.9945	0.995	0.998	0.9998	0.999	0.998	0.9993	0.999
		15	0.995	0.9952	0.995	0.998	0.9999	0.999	0.998	0.9997	0.999
		20	0.995	0.9961	0.996	0.999	0.9919	0.995	0.999	0.9987	0.999
		35	0.996	0.9987	0.997	0.999	0.9921	0.995	0.999	0.9983	0.999
		50	0.998	0.9983	0.998	0.999	0.9963	0.997	0.999	0.9981	0.998
		68	0.998	0.9985	0.998	0.999	0.9921	0.995	0.999	0.9982	0.999
	IG	10	0.996	0.9952	0.996	0.999	0.9919	0.995	0.999	0.9988	0.999
		15	0.996	0.9965	0.996	0.999	0.9921	0.995	0.999	0.9987	0.999
		20	0.997	0.9967	0.997	0.999	0.9920	0.995	0.999	0.9986	0.999
	IG										

Table A.2: Precision, Recall and F-Score of malware detection method with three feature selection techniques on all four generated datasets with eight different feature lengths and all three feature classes.

DS[a]	FS[b]	FL [c]	RBF-SVM			Random Forests			XGB		
			Prec [c]	Recall	F-1 [d]	Prec [c]	Recall	F-1 [d]	Prec [c]	Recall	F-1[d]
		35	0.998	0.9976	0.998	0.999	0.9966	0.998	0.999	0.9985	0.999
		50	0.998	0.9972	0.997	0.998	0.9920	0.995	0.999	0.9986	0.999
		68	0.998	0.9985	0.998	0.999	0.9977	0.998	0.999	0.9982	0.999
	FI	10	0.995	0.9955	0.995	0.999	0.9918	0.995	0.999	0.9990	0.999
		15	0.995	0.9954	0.995	0.999	0.9921	0.995	0.999	0.9993	0.999
		20	0.995	0.9981	0.997	0.999	0.9919	0.995	0.999	0.9987	0.999
		35	0.998	0.9985	0.998	0.999	0.9985	0.999	0.999	0.9980	0.999
		50	0.998	0.9987	0.998	0.999	0.9926	0.996	0.999	0.9982	0.999
		68	0.998	0.9985	0.998	0.999	0.9977	0.998	0.999	0.9982	0.999

[a] Selection of Dataset from Table 4.3

[b] Feature Selection Technique from Section 3.2.3

[c] Feature Length

[d] Precision

[e] F-Score