Detection of malicious RTF Files Using Machine Learning Methods

Liad Nahmias, Alex Lerman, Aviad Cohen, Nir Nissim, Yuval Elovici

Department of Information Systems Engineering, Ben-Gurion University of the Negev, Israel

Malware Lab, Cyber Security Research Center, Ben-Gurion University of the Negev, Beer-Sheva, Israel

TABLE OF CONTECTS:

I. ABSTRACT

In the last decade, we are witnessing an increasing in Cyber-attacks, hacking actions and exploitation of vulnerabilities. Hackers, espionage agencies and even countries use cyber-attacks in order to earn money, get private information and make damage to the victim's equipment. A detection of those Cyber-attacks could help the user to prevent leaking of private and important data.

One of the most common ways to launch a cyber-attack is using malicious files. Non-executable files considered safe by most of the users and enables the attacker to perform malicious actions in practice.

The existing detection methods e.g. Anti-Virus softwares, FireWalls etc. are not good enough for early detetcion of maicious files and zero-days attacks, especially, in RTF files.

In our Project we examined and demonstrated two main feature selection methods, using machine learning techniques, for early detection of malicious RTF and JPG files. The first one is based on the structure of the specific file while the second methods is based on learning in depth the structure of the general files structure, former attacks and vulnerabilities.

We conducted and compared the detection ratio of those methods in three different experiments. we showed that both of the methods are a good way for early detection that surpass known Anti-Virus softwares.

Keywords — RTF, Detection, Machine Learning, Analysis, Feature Extraction, knowledge-based, file structure

II. INTRODUCTION

In the last decade, we are witnessing an increasing in Cyber-attacks, hacking actions and exploitation of vulnerabilities. Hackers, espionage agencies and even countries use cyber-attacks in order to earn money, get private information and make damage to the victim's equipment.

One of the most common ways to launch a cyber-attack is using malicious files. Since executable files considered dangerous, attackers tend to leverage non-executable files in order to launch cyber-attacks. Non-executable files considered safe by most of the users and enables the attacker to perform malicious actions in practice.

Our project will be focusing on **Rich Text Format** (RTF) documents. RTF was crafted by Microsoft for creating text and graphic files with easy transfer between applications without dependecies on softwares and operation systems.

The detection of malicious RTF files is crucial. Attackers used this format many time and many vulnerabilities were discovered. One of the most famous cyber-attack used RTF file target the Indian Ambassador to Afghanistan[1]. On December 24, 2015 was sent an email consists crafted RTF file as attachments to the Indian ambassador to Afghanistan. The RTF file was a downloader of malware "Rover" that can take photos from the web camera of the victim machine, recording audio etc.

In this research we are using static analysis (examining the file without running it) in conjunction with machine learning (ML) in order to detect malicious RTF documents. We focus on the feature extraction process which aimed at extracting discriminative features from the examined documents in order to improve the detection capabilities of machine learning algorithms. In this research we propose two different approaches for feature extraction aimed at RTF documents. The first approach proposes a set of knowledge-based features. The second approach proposes a new feature extraction methodology that leverages the structure of the document and transform it to features. We extensively evaluate these two approches and compare them in a set of experiments.

The rest of the document is organizes as follows. We provide background information related to our research in Section 0III. Section E**Error! Reference source not found.** presents related work. In Section **Error! Reference source not found.** we describe the methods used in this research. Section 0 contains our evaluation and results. We discuss the results and security aspects, and present our conclusions, limitations, and future work in Section 3.

[1] https://philipcao.com/2016/03/01/new-malware-rover-targets-indian-ambassador-to-afghanistan/

III. BACKGROUND

RTF is an abbreviation for Rich Text Format file. Microsoft have crafted this format for creating text and graphic files with easy transfer between applications without dependecies on softwares and operation systems. For example, you don't need to have office softwares on your computer to open a RTF file. Morover, you can open this file on Mac and computers with linux. RTF is a text file (7-bit ASCII character) contains 4 main types of strings:

1. **Groups** - Contains control words or control symbols and text enclosed in curly braces { }. Each group specifies the text affected by the group and the different attributes of that text. RTF file can also include groups for:
   1.1 Fonts
   1.2 Styles
   1.3 Screen color
   1.4 Pictures
   1.5 Footnotes
   1.6 Comments (annotations)
   1.7 Headers and footers
   1.8 Summary information
   1.9 Fields
   1.10 Bookmarks
   1.11 Document-
   1.12 Section-
   1.13 Paragraph-
   1.14 Character-formatting properties

   Special cases:

   1.1 If the font, file, style, screen-color, revision mark, and summary-information groups and document-formatting properties are included, they must precede the first plain-text character in the document - These groups form the RTF file header.
   1.2 If the group for fonts is included, it should precede the group for styles.
   1.3 If any group is not used, it can be omitted.
   1.4 Formatting specified within a group affects only the text within that group. Generally, text within a group inherits the formatting of the text in the preceding group.
2. **Control words** - A command expression tells RTF how to handle the information inside it. A control word must start with '\' and cannot be longer than 32 characters. The structure of a control word:

   \CharactersSequence<Delimiter>

   For example: \creatim\yr1990 (create time year 1990)
   CharactersSequence - Made up characters between 'a' to 'z', lower case.
   Delimiter - The part that ends a control word, Can be:
   2.1 A space (the space is part of the control word)
   2.2 A digit or a hyphen(-), which indicates that a numeric parameter follows:
      2.2.1 There can be a space between the hyphen and the number.
      2.2.2 The parameter can be positive or negative.
      2.2.3 If the numeric parameter immediatlly follows the control word, it counts part of it.
   2.3 Any character other than a letter or a digit. (not count as part of ther control word)

Special cases:

2.1 Some of the control words properties have only two states.
  2.1.1 **\b** – bold is turn on
  2.1.2 **\b0** – bold is turn off
2.2 Control words can refer to a destination. They mark the beginning of a collection of related text that could appear at another position, or destination, within the document.

3. **Control symbols** - a control symbol represents a command. The structure of a control symbol:
   \NonalphabeticCharacter
   For example: \~ (a nonbreaking space)
   NonalphabeticCharacter - A single nonalphabetic character.
   Special cases:
   3.1 ***** - Identifies destinations whose related text should be ignored if the RTF reader does not recognize the destination.
4. **Plain text** - An unformatted lower case english letters, digits and symbols (e.g. ~,%, * etc.) that are part of the main group only. Figure 1 presents an example of plain text. Figure 2 presents an RTF file and part of its code.

```
{\rtf1\ansi\deff0{\fonttbl{\f0\froman Tms Rmn;}{\f1\fdecor Symbol;}{\f2\fswiss Helv;}}
{\colortbl;\red0\green0\blue0;\red0\green0\blue255;\red0\green255\blue255;\red0\green255\blue0;\red25
5\green0\blue255;\red255\green0\blue0;\red255\green255\blue0;\red255\green255\blue255;}
{\stylesheet{\fs20 \snext0 Normal;}}{\info{\author John Doe}{\creatim\yr1990\mo7\dy30\hr10\min48}
{\version1}{\edmins0}{\nofpages1}{\nofwords0}{\nofchars0}{\vern8351}}
\widoctrl\ftnbj \sectd\linex0\endnhere \pard\plain \fs20 This is plain text.\par}
```

Figure 1 The phrase "This is plain text" is not part of an inside group (related to \rtf1 group) and is treated as document text.

```
{\rtf1\adeflang1037\ansi\ansicpg1252\uc1\adeff31507
{\f37\fbidi \fswiss\fcharset0\fprq2{\*\panose 020f0
{\fdbmajor\f31501\fbidi \froman\fcharset0\fprq2{\*\
{\fbimajor\f31503\fbidi \froman\fcharset0\fprq2{\*\
{\fdbminor\f31505\fbidi \froman\fcharset0\fprq2{\*\
{\fbiminor\f31507\fbidi \fswiss\fcharset0\fprq2{\*\
{\f59\fbidi \fswiss\fcharset162\fprq2 Arial Tur;}{\
{\f63\fbidi \fswiss\fcharset163\fprq2 Arial (Vietna
{\f60\fbidi \fswiss\fcharset177\fprq2 Arial (Hebrew
{\f415\fbidi \fswiss\fcharset238\fprq2 Calibri CE;}
```

This is a RTF File.

1. Sample
2. Example
   a. Testing
   b. Testing

ID	Name

Figure 2. RTF file and part of its source code.

A. RTF File Structure

The RTF structure is simple and contains two main parts: Header and Document in the following order: **{<header> <document>}**. Figure 3 presents a scheme of RTF file structure.

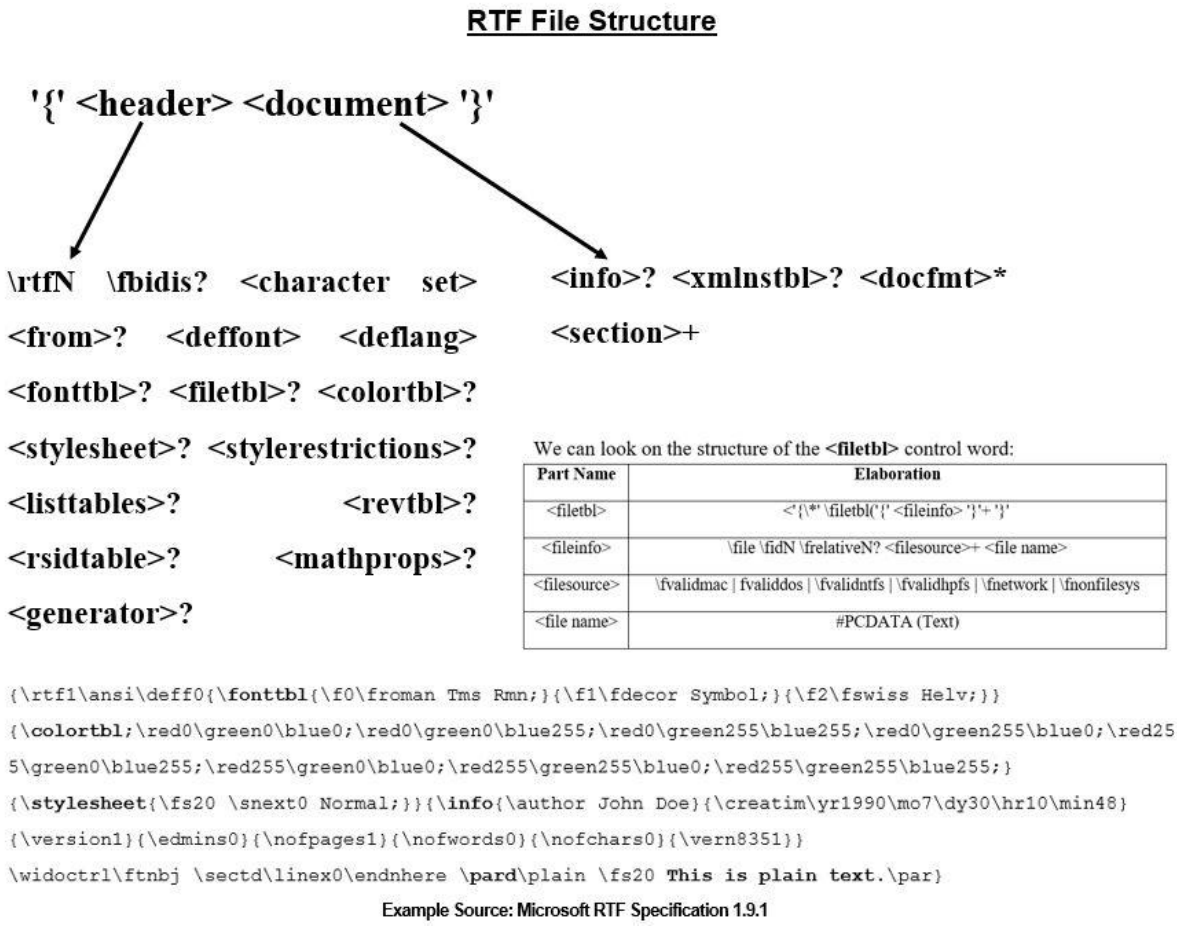


Figure 3. RTF file structure and an example.

1. **Header** -

   The header part of the file contains all the meta data needed to the document e.g. styles, fonts colors etc.

   The header has a very specific syntax:

   \rtfN \fbidis? <character set> <from>? <deffont> <deflang> <fonttbl>? <filetbl>? <colortbl>? <stylesheet>? <stylerestrictions>? <listtables>? <revtbl>? <rsidtable>? <mathprops>? <generator>?

   Each part has its specific role. Table 1 presents the header's main control words and their role as meta data.

Header's parts	Appearance	Role
\rtfN	Mandatory	Magic number (Identifier) of RTF file. N is the version number (supposed to be 1).

\fbidis	Optional	Indicates a single font is active instead of a set of associated fonts.
<character set>	Mandatory	Default characters set used in the document.
<from>	Optional	The original source of the document (html, email etc.).
<deffont>	Mandatory	Default font.
<deflang>	Mandatory	Default language.
<fonttbl>	Optional	Control word introduces the font table group.
<filetbl>	Optional	Control word introduces the file table destination.
<colortbl>	Optional	Control word introduces the color table group, which defines screen colors, character colors, and other color information.
<stylesheet>	Optional	Control word introduces the style sheet group, which contains definitions and descriptions of the various styles used in the document.
<stylerestrictions>	Optional	Define the restrictions for the style.
<listtables>	Optional	Define the list tables in the document. There are two type of list tables: list table and list override table
<revtbl>	Optional	Table tracking of multiple authors and reviewers of the document.
<rsidtable>	Optional	Table that reserved all the changes in the document.
<mathprops>	Optional	Contains the math properties e.g. bar, matrix etc.
<generator>	Optional	Stamp the document (name, author etc.)

Table 1. Header's Syntax parts

Each part of the Header Syntax consists more parts and control words.

2. **Document -**
   The document part of the file contains all the information of the document.
   The document has a very specific syntax:
   <info>?<xmlnstbl>?<docfmt>* <section>+

   Each part has its specific role. Table 2 presents the document's syntax parts and their role as data.

Document's parts	Appearance	Role
<info>	Optional	Control word introduces the information group, which contains information about the document.
<xmlnstbl>	Optional	XML namespace and SmartTags are used in an RTF formatted document.
<docfmt >	Zero or more	Control words specify the document attributes (margin, footnotes etc.)

<section>	One or more	Contains the data of the document.

Table 2. Document's syntax parts

B. Attack Techniques

As we mentioned in the introduction, we are witness growth in hacking action all over the globe. Everyday hackers find new techniques or new ways to improve their former version of codes to succeed infecting more machines. RTF files were used in many different attacks and many vulnerabilities were exposed. The main techniques used RTF files are summarized below:

1. ***OLE/Embedded objects*** [2] **-** The RTF file structure allows storing OLE objects with potentially malicious content, the OLE package objects can store any kind of file including executables and scripts. When a user clicks on and OLE object the embedded file will be opened by the system, this feature is used to deliver malware. The OLE objects are stored using several nested control words ({\object..{objdata..}}). The object data is serialized to a byte string and that string is encoded in hexadecimal digits with optional spaces and new line characters to split long lines.

2. ***Remote code execution***[3] **-** The ability to trigger arbitrary code from one machine on another. An attacker ability to execute any command of the attacker's choice on the target machine or the target process. In early 2016 a research group discovered a vulnerability in Microsoft office that fails to parse a RTF file which is a heap overflow vulnerability. The research group analyzed a sample of a crafted RTF and discovered that the attacker used an error while handling the "pict" structure which allows RTF file to include pictures in hexadecimal or binary format created with other applications using the control word "\\pict". The crafted file contained a difference. (Change of a bit in the picture format data which causes a construction of a smaller heap buffer which causes a heap overflow when copying data).

3. ***Exploiting known vulnerabilities in not up to date softwares*** **-** Many of users don't update their softwares and computers. This can lead to use known vulnerabilities to hack their machines based on RTF files even if there is a known way how to prevent this hacking action.

4. ***Obfuscation techniques of malicious RTF files***[4] **-** Manipulating the RTF structure to evade static analysis of the file. There are many ways that attacker use to cruft the RTF file so it won't be detected by analysis tools that use signature based static analysis to hide malicious content like malicious embedded files by adding brackets or whitespaces between the binary data of the embedded file.

5. ***Embedded files dropped into temp folder*** [5] **-** "When a RTF formatted document is opened that includes embedded objects (objects inserted from a file), the objects are extracted to the user's temp directory where they are launched with the default handler if the user clicks the object within the document. Once the document is closed, the files removed from the user's temp

---

[2] https://securelist.com/analysis/publications/37158/the-curious-case-of-a-cve-2012-0158-exploit/, http://www.sekoia.fr/blog/ms-office-exploit-analysis-cve-2015-1641/

[3] https://blog.fortinet.com/2016/01/20/deep-analysis-of-cve-2016-0010-microsoft-office-rtf-file-handling-heap-overflow-vulnerability

[4] https://www.fireeye.com/blog/threat-research/2016/05/how_rtf_malware_evad.html, http://www.decalage.info/rtf_tricks, https://www.ixiacom.com/company/blog/malware-delivery-secrets-rtf-obfuscation

[5] https://www.trustwave.com/Resources/SpiderLabs-Blog/Analysis-of-Malicious-Document-Files-Spammed-by-Cutwail/, http://www.sekoia.fr/blog/ms-office-exploit-analysis-cve-2015-1641/, http://phishme.com/rtf-malware-delivery/

directory. During the time period the document is open, these files are available to other processes on the system".

C. List of CVEs

Vulnerability is a weakness in an application that arises from the way the application is written. A vulnerability in an application could allow an attacker to compromise the integrity, availability, or confidentiality of the application. CVE[6] (Common Vulnerabilities and Exposures) is a standard for information security vulnerability names. The MITRE Corporation maintains the CVE system which is a free, publicly available dictionary of known information security vulnerability and exposures, subsidized by the National Cyber Security Division of United States Department of Homeland Security. The CVE-ID format is: "CVW-YYYY-NNNN" (YYYY – 4-digit Year, NNNN – 4-digit number). Table 3 presents a list of known CVEs for JPEG images.

ID	CVE Name	Description	Effect
1	CVE-2016-7193	Microsoft word 2007 SP2, Office 2010 SP2, Word 2013 SP1, Word 2013 RT SP1, Word 2016, Word for Mac 2011, Word 2016 for Mac, Office Compatibility Pack SP3, Word Viewer, Word Automation Services on SharePoint Server 2010 SP2, Word Automation Services on SharePoint Server 2013 SP1, Office Web Apps 2010 SP2, Office Web Apps 2013 SP1 and Office Online Server allow remote attackers to execute arbitrary code via a crafted RTF document, aka "Microsoft Office Memory Corruption Vulnerability."[7]	An attacker can leverage this issue to execute arbitrary code in the context of the currently logged-in user.[8]
2	CVE-2015-1641	Microsoft word 2007 SP3, Office 2010 SP2,Word 2010 SP2, Word 2013 SP1, Word 2013 RT SP1, Word for Mac 2011, Office Compatibility Pack SP3, Word Automation Services on SharePoint Server 2010 SP2, Word Automation Services on SharePoint Server 2013 SP1, Office Web Apps Server 2010 SP2 and Office Web Apps Server 2013 SP1 allow remote attackers to execute arbitrary code via a crafted RTF document, aka "Microsoft Office Memory Corruption Vulnerability."[9]	An attacker can leverage this issue to execute arbitrary code in the context of the currently logged-in user.[10]
3	CVE-2015-0086	Microsoft word 2007 SP3, Office 2010 SP2, Word 2010 SP2, Word 2013 SP1, Word 2013 Gold, Word 2013 RT SP1, Word 2013 RT Gold, Word Viewer, Office Compatibility Pack SP3, Word Automation Services on SharePoint Server 2010 SP2, Word Automation Services on SharePoint Server 2013 SP1,	An attacker can leverage this issue to execute arbitrary code in the context of the currently logged-in

[6] http://cve.mitre.org/
[7] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-7193
[8] https://www.symantec.com/security_response/vulnerability.jsp?bid=93372
[9] https://.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-1641
[10] https://www.symantec.com/security_response/vulnerability.jsp?bid=73995

		Word Automation Services on SharePoint Server 2013 Gold, Web Applications 2010 SP2, Web Apps Server 2013 SP1 and Web Apps Server Gold allow remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted RTF document, aka "Microsoft Office Memory Corruption Vulnerability."[11]	user.[12]
4	CVE-2014-1761	Microsoft Word 2003 SP3, Word 2007 SP3 , Word 2010 SP1, Word 2010 SP2, Word 2013 SP1, Word 2013 RT SP1, Word Viewer, Office Compatibility Pack SP3, Office for Mac 2011, Word Automation Services on SharePoint Server 2010 SP1, Word Automation Services on SharePoint Server 2010 SP2,Word Automation Services on SharePoint Server 2013 SP1, Office Web Applications 2010 SP1, Office Web Apps 2010 SP2, and Office Web Apps Server 2013 allow remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted RTF data, as exploited in the wild in March 2014.[13]	An attacker can leverage this issue to execute arbitrary code in the context of the currently logged-in user.[14]
5	CVE-2012-2539	Microsoft Word 2003 SP3, Word 2007 SP2 , Word 2007 SP3 , Word 2010 SP1, Word Viewer, Office Compatibility Pack SP2, Office Compatibility Pack SP3 and Office Web Apps 2010 SP1 allow remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted RTF data, aka "word RTF "listoverridecount" Remote Code Execution Vulnerability."[15]	An attacker who successfully exploited the vulnerability could gain the same user rights as the current user.[16]
6	CVE-2012-2528	Use-after-free vulnerability in Microsoft Word 2003 SP3, Word 2007 SP2, Word 2007 SP3, Word 2010 SP1, Word Viewer, Office Compatibility Pack SP2, Office Compatibility Pack SP3, Word Automation Services on Microsoft SharePoint Server 2010 and Office Web Apps 2010 SP1 allow remote attackers to execute arbitrary code via a crafted RTF document, aka "RTF File listid User-After-Free Vulnerability."[17]	An attacker who successfully exploited the vulnerability could gain the same user rights as the current user.[18]

[11] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0086
[12] https://www.symantec.com/security_response/vulnerability.jsp?bid=72911
[13] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1761
[14] https://www.symantec.com/connect/blogs/emerging-threat-microsoft-word-zero-day-cve-2014-1761-remote-code-execution-vulnerability
[15] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2539
[16] https://www.rapid7.com/db/vulnerabilities/WINDOWS-HOTFIX-MS12-079
[17] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2528

7	CVE-2012-0183	Microsoft Word 2003 SP3, Word 2007 SP2, Word 2007 SP3, Office 2008, Office 2011 for Mac, Office Compatibility Pack SP2 and Office Compatibility Pack SP3 allow remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via crafted RTF data, aka "RTF Mismatch Vulnerability."[19]	An attacker can exploit this issue to execute arbitrary code in the context of the currently logged-in user.[20]
8	CVE-2012-0158	The (1) ListView, (2) ListView2, (3) TreeView and (4) TreeView2 ActiveX controlos in MSCOMCTL.OCX in the Common Controls in Microsoft Office 2003 SP3, Office 2007 SP2, Office 2007 SP3, Office 2010 SP1, Office 2010 Gold, Office 2003 Web Components SP3, SQL Server 2000 SP4, SQL Server 2005 SP4 and SQL Server 2008 SP2, SQL Server 2008 SP3, SQL Server 2008 R2, BizTalk Server 2002 SP1, Commerce Server 2002 SP4, Commerce Server 2007 SP2, Commerce Server 2009 Gold, Commerce Server 2009 R2, Visual FoxPro 8.0 SP1, Visual FoxPro 9.0 SP2 and visual Basic 6.0 Runtime allow Remote attackers to execute arbitrary code via a crafted (a) web site, (b) Office document or (c) .rtf file that triggers "system state" corruption, as exploited in the wild in April 2012, aka "MSCOMCTL.OCX RCE Vulnerability."[21]	An attacker can exploit this issue by enticing an unsuspecting user to view a malicious webpage. Successful exploits will allow the attacker to execute arbitrary code within the context of the application (typically Internet Explorer) that uses the ActiveX control.[22]
9	CVE-2010-3333	Stack-based buffer overflow in Microsoft Office XP SP3, Office 2003 SP3, Office 2007 SP2, Office 2010, Office 2004, Office 2008 for Mac, Office for Mac 2011, Open XMK File Format Converter for Mac allow remote attackers to execute arbitrary code via crafted RTF data, aka "RTF Stack Buffer Overflow Vulnerability."[23]	An attacker can exploit this issue by enticing an unsuspecting user to open a malicious RTF file or view an email in RTF format. Successfully exploiting this issue would allow the attacker to corrupt memory and execute arbitrary code

[18] https://www.symantec.com/security_response/vulnerability.jsp?bid=76590
[19] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0183
[20] https://www.symantec.com/security_response/vulnerability.jsp?bid=53344
[21] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0158
[22] https://www.symantec.com/security_response/attacksignatures/detail.jsp?asid=25656
[23] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-3333

			in the context of the currently logged-in user.[24]
10	CVE-2010-1902	Buffer overflow in Microsoft Office Word 2002 SP3, Office Word 2003 SP3, Office Word 2007 SP2, Microsoft office 2004 for Mac, Microsoft office 2008 for Mac, Open XMK File Format Converter for Mac, Office Word Viewer, Office Compatibility Pack for Word, Excel and PowerPoint 2007 File Format SP2 allow remote attackers to execute arbitrary code via unspecified properties in the data in a crafted RTF document, aka "Word RTF Parsing Buffer Overflow Vulnerability."[25]	An attacker who successfully exploited any of these vulnerabilities could gain the same user rights as the local user.[26]
11	CVE-2010-1901	Microsoft Office Word 2002 SP3, Word 2003 SP3, Word 2007 SP2, Microsoft Office 2004 for Mac, Microsoft Office 2008 for Mac, Open XML File Format Converter for Mac, Office Word Viewer, Office Compatibility Pack for Word, Excel and PowerPoint 2007 File Formats SP2 do not properly handle unspecified properties in rich text data, which allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted RTF document, aka "Word RTF Parsing Engine Memory Corruption Vulnerability."[27]	An attacker who successfully exploited any of these vulnerabilities could gain the same user rights as the local user.[28]
12	CVE-2008-4031	Microsoft Office Word 2000 SP3, Word 2002 SP3, Word 2003 SP3, Word 2007 Gold, Word 2007 SP1, Outlook 2007 Gold, Outlook 2007 SP1, Word Viewer 2003 Gold, Word Viewer 2003 SP3, Office compatibility Pack for Word, Excel and PowerPoint 2007 File Format Gold and SP1, Office 2004 for Mac, Office 2008 for Mac and Open XML File Format Converter for Mac allow remote attackers to execute arbitrary code via a malformed string in (1) an RTF file or (2) a rich text e-mail message, which triggers incorrect memory allocation and memory corruption, aka "Word RTF Object Parsing Vulnerability."[29]	An attacker who successfully exploited these vulnerabilities could take complete control of an affected system. An attacker could then install programs, view, change or delete data or create new accounts with full user rights.[30]

[24] https://www.symantec.com/security_response/attacksignatures/detail.jsp?asid=23997
[25] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-1902
[26] https://www.rapid7.com/db/vulnerabilities/WINDOWS-HOTFIX-MS10-056
[27] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-1901
[28] https://www.rapid7.com/db/vulnerabilities/WINDOWS-HOTFIX-MS10-056
[29] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4031
[30] https://www.rapid7.com/db/vulnerabilities/WINDOWS-HOTFIX-MS08-072

13	CVE-2008-4030	Microsoft Office Word 2000 SP3, Word 2002 SP3, Word 2003 SP3, Word 2007 Gold, Word 2007 SP1, Outlook 2007 Gold, Outlook 2007 SP1, Word Viewer 2003 Gold, Word Viewer 2003 SP3, Office compatibility Pack for Word, Excel and PowerPoint 2007 File Format Gold and SP1, Office 2004 for Mac, Office 2008 for Mac and Open XML File Format Converter for Mac allow remote attackers to execute arbitrary code via a malformed string in (1) an RTF file or (2) a rich text e-mail message, which triggers incorrect memory allocation and memory corruption, aka "Word RTF Object Parsing Vulnerability." A different vulnerability than CVE-2008-4028.[31]	An attacker who successfully exploited these vulnerabilities could take complete control of an affected system. An attacker could then install programs, view, change or delete data or create new accounts with full user rights.[32]
14	CVE-2008-4028	Microsoft Office Word 2000 SP3, Word 2002 SP3, Word 2003 SP3, Word 2007 Gold, Word 2007 SP1, Outlook 2007 Gold, Outlook 2007 SP1, Word Viewer 2003 Gold, Word Viewer 2003 SP3, Office compatibility Pack for Word, Excel and PowerPoint 2007 File Format Gold and SP1, Office 2004 for Mac, Office 2008 for Mac and Open XML File Format Converter for Mac allow remote attackers to execute arbitrary code via a malformed string in (1) an RTF file or (2) a rich text e-mail message, which triggers incorrect memory allocation and memory corruption, aka "Word RTF Object Parsing Vulnerability." A different vulnerability than CVE-2008-4030.[33]	An attacker who successfully exploited these vulnerabilities could take complete control of an affected system. An attacker could then install programs, view, change or delete data or create new accounts with full user rights.[34]
15	CVE-2008-4027	Double free vulnerability in Microsoft Office Word 2000 SP3, 2002 SP3, 2003 SP3 and 2007 Gold and SP1, Outlook 2007 Gold and SP1, Word viewer 2003 Gold and SP3, Office Compatibility Pack for Word, Excel and PowerPoint 2007 File Formats Gold and SP1 and Office 2004 for Mac allow remote attackers to execute arbitrary code via a crafted (1) RTF file or (2) rich text e-mail message with multiple consecutive Drawing Object ("\do") tags, which triggers a "memory calculation error" and memory corruption, aka "Word RTF Object Parsing Vulnerability."[35]	An attacker who successfully exploited these vulnerabilities could take complete control of an affected system. An attacker could then install programs, view, change or delete data or create new accounts with full user rights.[36]

[31] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4030
[32] https://www.rapid7.com/db/vulnerabilities/WINDOWS-HOTFIX-MS08-072
[33] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4028
[34] https://www.rapid7.com/db/vulnerabilities/WINDOWS-HOTFIX-MS08-072
[35] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4027
[36] https://www.rapid7.com/db/vulnerabilities/WINDOWS-HOTFIX-MS08-072

16	CVE-2008-4025	Integer overflow in Microsoft Office Word 2000 SP3, 2002 SP3, 2003 SP3 and 2007 Gold and SP1, Outlook 2007 Gold and SP1, Word Viewer 2003 Gold and SP3, Office Compatibility Pack for Word, Excel and PowerPoint 2007 File Formats Gold and SP1, Office 2004 and 2008 for Mac and Open XML File Format Converter for Mac allow remote attackers to execute arbitrary code via (1) an RTF file or (2) a rich text e-mail message containing an invalid number of points for a polyline or polygon, which triggers a heap-based buffer overflow, aka "word RTF Object Parsing Vulnerability."[37]	An attacker who successfully exploited these vulnerabilities could take complete control of an affected system. An attacker could then install programs, view, change or delete data or create new accounts with full user rights.[38]
17	CVE-2008-1091	Unspecified vulnerability in Microsoft Word in Office 2000 and XP SP3, 2003 SP2 and SP3 and 2007 Office System SP1 and earlier allow remote attackers to execute arbitrary code via Rich Text Format (.rtf) file with a malformed string that triggers a "memory calculation error" and a heap-based buffer overflow, aka "Object Parsing Vulnerability."[39]	An attacker who successfully exploited these vulnerabilities could take complete control of an affected system. An attacker could then install programs, view, change or delete data or create new accounts with full user rights.[40]
18	CVE-2007-1202	Word (or Word Viewer) in Microsoft Office 2000 SP3, XP SP3, 2003 SP2, 2004 for Mac and Works Suite 2004, 2005 and 2006 do not properly parse certain rich text "property strings of certain control words" which allows user-assisted remote attackers to trigger heap corruption and execute arbitrary code, aka the "Word RTF Parsing Vulnerability."[41]	An attacker could exploit the vulnerability by constructing a specially crafted Word file that could allow remote code execution. An attacker who successfully exploited this vulnerability could gain the same user rights as the local user.[42]

[37] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4025
[38] https://www.rapid7.com/db/vulnerabilities/WINDOWS-HOTFIX-MS08-072
[39] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-1091
[40] https://www.rapid7.com/db/vulnerabilities/WINDOWS-HOTFIX-MS08-026
[41] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-1202
[42] https://technet.microsoft.com/en-us/library/security/ms07-024.aspx

19	CVE-2006-1311	The RichEdit component in Microsoft Windows 2000 SP4, XP SP2 and 2003 SP1, Office 2000 SP3, XP SP3, 2003 SP2 and Office 2004 for Mac and Learning Essentials for Microsoft Office 1.0, 1.1 and 1.5 allow user-assisted remote attackers to execute arbitrary code via a malformed OLE object in a RTF file, which triggers memory corruption.[43]	An attacker who successfully exploited these vulnerabilities could take complete control of an affected system. An attacker could then install programs, view, change or delete data or create new accounts with full user rights.[44]
20	CVE-2004-0848	Buffer overflow in Microsoft Office XP allows remote attackers to execute arbitrary code via a link with a URL file location containing long inputs after (1) "%00" (null byte) in .doc filenames or (2) "%0A" (carriage return) in .rtf filenames.[45]	An attacker who successfully exploited this vulnerability could allow execute arbitrary code via a link with a URL file location containing long inputs after (1) "%00" (null byte) in .doc filenames or (2) "%0A" (carriage return) in .rtf filenames.[46]

Table 3. CVE list for RTF files.

[43] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-1311
[44] https://www.rapid7.com/db/vulnerabilities/WINDOWS-HOTFIX-MS07-013
[45] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-0848
[46] https://cve.circl.lu/cve/CVE-2004-0848

D. Analysis and Detetion Tools

There are many detection tools available on the web. Every one of them detects malicious activities in different formats of files (only for Image, only office documents etc.) and used different ways (Static analysis/Dynamic analysis/both). For comparison usage, we summarized all the existing tools suggest detection for malicious RTF files:

1. **RTFScan OfficeMalScanner suit -** Scans RTF files for malicious traces like shellcode heuristics, dump object and data areas, as well as PE(Portable Executable)-Files.
2. **rtfobj oletools -** A python tool to extract embedded objects from RTF files.
3. **psparser** [47] **-** Processes the data of an embedded object to extract the metadata and optionally the embedded file itself.
4. **O-checker** [48] **-** Detection of malicious documents through deviation from file format.
5. **VirusTotal**[49] **-** A free web service that analyzes suspicious files, URLs and checking them with many Anti-virus softwares (more than 60). The service can detect viruses, worms, Trojans and all kind if malware and provides to the user a report with verdict.

E. Approaches for Malware Detection

There are two fundamental methods for malware detection, the first is dynamic and the second is static. The main difference between those two methods is that in static analysis we don't run the code while in dynamic analysis we do run it.

Those two methods let you check in real-time the file and detect the malicious activities and code that can harsh your machine.

Both of them have cons and pros:

Static analysis will not run the malware so you can determinate the code on your own machine while using dynamic analysis will be required a virtual machine or other machine.

On the other hand, when you need to use static analysis on an obfuscated and big size of code, Dynamic analysis will help you just run it and see the code behavior, activities on the machine etc.

Another known method for malware detection is based on the file signature. File signature means to take the file and calculated his hash (SHA-1, SHA-256, MD5 etc.). In this way you create a unique identifier to the file. Anti-Virus softwares use this way and it works pretty well but this way has its own big disadvantage – if the Anti-Virus software has to give verdict on Zero-Day attack or even on a file it has in her Database but with a change of a character, this methods couldn't work well.

---

[47] https://github.com/phishme/malware_analysis/blob/master/scripts/psparser.py
[48] https://www.blackhat.com/docs/us-16/materials/us-16-Otsubo-O-checker-Detection-of-Malicious-Documents-through-Deviation-from-File-Format-Specifications-wp.pdf
[49] http://www.virustotal.com/

IV. RELATED WORK

In this section we present papers related to our research topic. The various articles and researches below helped us to choose our methodology of analysis (between static and dynamic) and the research work flow - Structural (aka feature extraction based on the file) and Knowledge-based (aka feature extraction based on former attacks and the file structure).

Ford, Cova, Kruegel and Kinda [FCKV00] showed the importance of both static and dynamic analysis when dealing with analysis and detection of malicious Flash advertisements. The authors created a tool calls "OdoSwiff" using static analysis to parse the tags of the flash. The parsing action is helping to detect known malicious techniques e.g. hiding malicious code (Shellcode or ActionScript code) and for finding a use of known vulnerabilities. "OdoSwiff" also uses dynamic analysis required to execute the Flash application and follows the execution trace in order to find out referenced URLs and network activities. For testing the success rate of "OdoSwiff", the authors collected corpuses of malicious Flash advertisements and compare the results to the detection rate of VirusTotal (a web platform that shows results of running files in many different Anti-Virus softwares). The results show that "OdoSwiff" got a higher detection rate, by 14% more than VirusTotal. Figure 4 presents those results of detection malicious flash advertisements out of 2,492 advertisements ("OdoSwiff" and "adopstools" produced one FP and "OdoSwiff" detected four malicious advertisements while "adopstools" detected three malicious advertisements.).

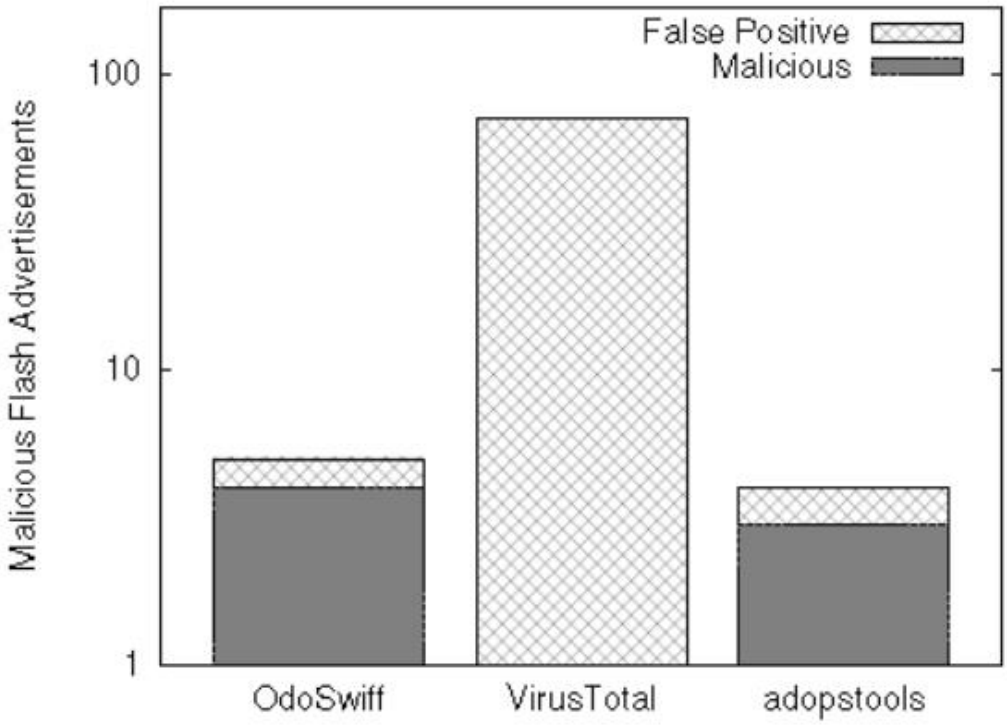


Figure 4. Detection results for crawled Flash advertisements

Article [JoKK00], deals with the vulnerabilities in web applications. The authors elaborated about the different vulnerabilities (SQL injection, cross-site scripting etc.) and created a tool named "Pixy". "Pixy" is based on static analysis that detects the malicious activities from the source code of the

applications. During the experiments, "Pixy" detected 15 unknown vulnerabilities in three web applications.

Christodorescu and Jha [Chri00] developed in their article a static analysis tool called "SAFE" for detecting malicious executable files. "SAFE" was created to provide a better solution for obfuscation attacks. After the experiments phase, the results showed that "SAFE" detected malicious files better than three familiar Anti-Virus software. All three succeed in detecting the different viruses but when obfuscation code was used, it tricked the Anti-Virus softwares.

In the article [ThSc13] Thesis and Science tried to check if it is possible to detect malicious PDF files which were sent as attachments via email. The author elaborated about malicious file that was sent as attachment, which can also be Microsoft office and EXE program. Through the research, the author used two tools - "PDF Scrutinizer" and "MDScan". Both of the tools use static and dynamic analysis of PDF files. The conclusion of the author was that it is possible to detect malicious PDF files but it is important to keep developing new technologies and tools to detect new malicious attacks.

The article [PrST16] focuses on steganography and how to detect a malware that has been deployed into an image. The authors built a tool for detecting malware from a stegno-image by using steganalysis calculations (chi-square assaults, visual discovery and histogram investigation). The authors concluded that the tool recognize some of the methods of steganography.

In article [HaVi00] Hallaraker and Vigna showed how to detect malicious JavaScript code in the Mozilla browser. They showed two ways to do it - the first is by anomaly detection and the second is by writing a signatures. Anomaly detection is the action of comparing the behavior of the script to a "normal" behavior, while writing signature is comparing the actions of the script to predefined attack scenarios. Example for writing signatures: every script that will open more than 4 windows or if the script set the location of the document with more than one argument, the system can detect and specify the script as malicious. This research shows the importance of using expert features in the detection efforts.

In article [Otsu15] Otsubo presented how experts can help to detect malicious file. By understanding the problem, the way of attack and how the attackers exploit the documents' vulnerabilities, the authors created a tool called "O-checker". In Japan, through 2014, many organizations had received targeted emails with malicious files as attachments. "O-checker" was created to help those organizations detect malicious files like PDF, RTF etc. The base of "O-checker" is that by understanding the structure of the files, it can define exceptions like embedded text after "EOF" or the correctness of the structure of the specific file. Through the experiments, the authors compared "O-checker" performance to Anti-Virus software. "O-checker" detected malicious files with 96.1% of success.

In article [MaGi00] Maiorca and Giacinto showed the simple way, in which attackers can change their malware code and create PDF files that can evade software that used structural features. The attack called "The Mimicry attack" when the hacker mimics a benign PDF structure and change the parts he knows the detection software is not checking. This article empowers the claim that detection efforts must keep developing all the time in order to be unpredictable and hard to bypass.

In article [ANWN07] a comparison between some of those techniques like Random forests, SVM, Neural Networks, Bayesian additive regression trees, logistic regression and Classification and Regression trees. For the experiments step, the author created a data set of malicious and cleaned emails (1171 raw phishing emails and 1718 legitimate emails) and parsed them for testing. After that,

they ran all the different models, 30 times with 43 features. The article's results show that Random forests presented the little errors but showed the worst FP ratio. Figure 5 presents the average error for all classifiers and we can see that Random Forest produces the little error. Figure 6 presents FP and FN ratio for all classifiers and we can see that Random Forest produces the worst FP ratio.

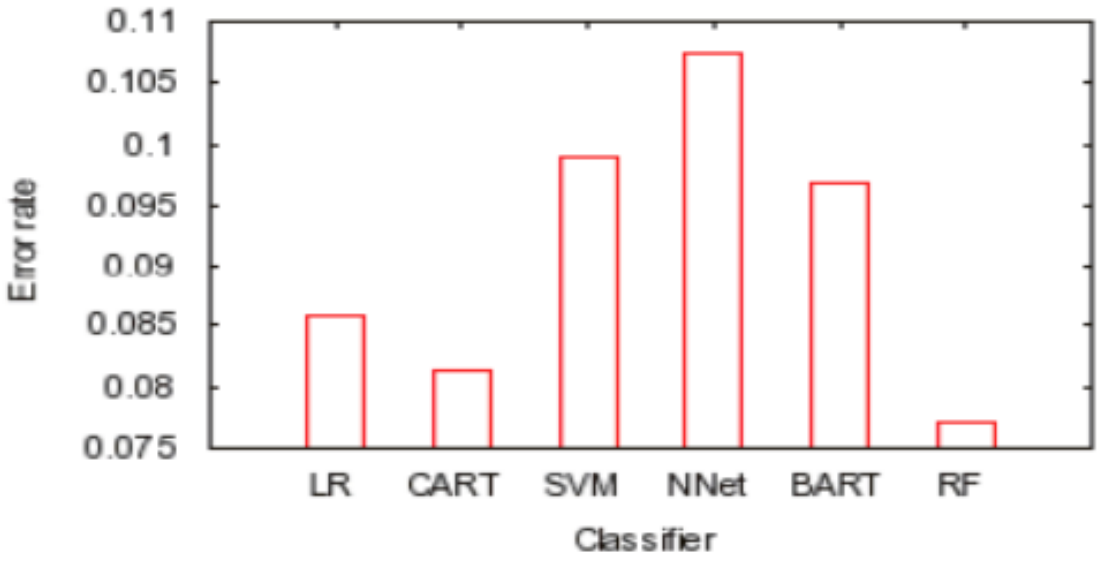


Figure 5. Average $W_{err}(\lambda = 1)$ for all clasifiers

	FP	FN
LR	04.89%	17.04%
CART	07.68%	12.93%
SVM	07.92%	17.26%
NNet	05.85%	21.72%
BART	05.82%	18.92%
RF	08.29%	11.12%

Figure 6. False positive (FP) rate and false negative (FN) rate for all classifiers

In article [WeKa89] Weiss and Kapouleas showed an empirical comparison between Pattern recognition (Linear discriminant, Quadratic discriminant, Nearest neighbor, Bayes independence and Bayes second order), neural nets and machine learning classification methods. To give a verdict, the authors ran four experiments (we will share the results of two of them). The first experiment consists of the Iris Database. Figure 7 presents the errors for all the classifiers for comparison usage while Figure 8 presents the rate of errors only for neural net on Fisher's Iris data.

Method	Err_{App}	Err_{Cv}
Linear	.020	.020
Quadratic	.020	.027
Nearest neighbor	.000	.040
Bayes independence	.047	.067
Bayes 2nd order	.040	.160
Neural net (BP)	.017	.033
PVM rule	.027	.040
Optimal rule size 2	.020	.020
CART tree	.040	.047

Figure 7.Comparative performance on Fisher's Iris data

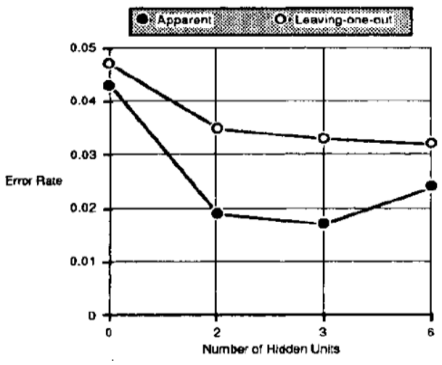


Figure 8. Neural net Error rates for Iris data

The second experiment consists of the Appendicitis Patients Database (106 patients and 8 diagnostic tests). Figure 9 presents the errors for all the classifiers for comparison usage while Figure 10 presents the rate of errors only for neural net on Appendicitis data. The authors didn't give a clear verdict, but they showed that before choosing the classifier, it is important to check some of them on the relevant database.

Method	Err_{App}	Err_{Cv}
Linear	.113	.132
Quadratic	.217	.264
Nearest neighbor	.000	.179
Bayes independence	.113	.170
Bayes 2nd order	.047	.189
Neural net (BP)	.098	.142
PVM rule	.085	.104
Optimal rule size 2	.085	.104
CART tree	.094	.151

Figure 9. Comparative performance on Appendicitis data

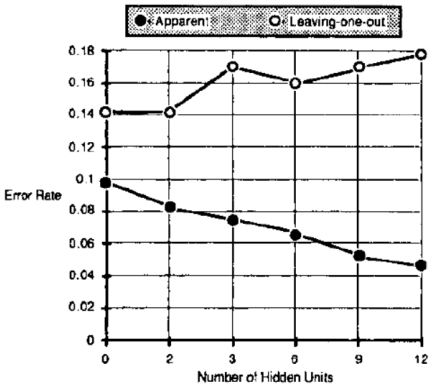


Figure 10. Neural net error rates for Appendicitis data

The article [ACKS00] focused on using the N-gram algorithm for the detection of new malicious code. According to this article, using N-gram has helped to detect strings and substrings that define malicious code and it evaluated the detection of malicious code to 98% at the test set. Figure 11 presents the accuracy for L (profile length) and n (n-gram size). We can see higher accuracy ratio as long as the authors used higher n and L.

	n									
L	1	2	3	4	5	6	7	8	9	10
20	0.71	0.77	0.70	0.79	0.81	0.80	0.77	0.82	0.85	0.85
50	0.88	0.76	0.73	0.86	0.88	0.74	0.83	0.82	0.85	0.83
100	0.91	0.88	0.74	0.77	0.88	0.83	0.89	0.88	0.83	0.89
200	0.91	0.94	0.76	0.76	0.91	0.91	0.91	0.88	0.88	0.86
500	0.79	0.97	0.89	0.76	0.92	0.95	0.89	0.86	0.89	0.88
1000	0.79	0.97	0.97	0.94	0.97	0.95	0.92	0.88	0.92	0.94
1500	0.79	0.95	0.98	0.97	0.98	0.97	0.94	0.91	0.91	0.94
2000	0.79	0.92	0.98	0.97	0.98	0.98	0.94	0.94	0.95	0.95
3000	0.79	0.94	0.98	0.97	0.97	0.97	0.97	0.95	0.97	0.98
4000	0.79	0.92	0.97	0.97	0.97	0.97	0.98	0.97	0.97	0.97
5000	0.79	0.92	0.97	0.95	0.94	0.95	0.98	0.98	0.97	0.97

Figure 11. 3-fold cross-validation accuracy

Another article [SPDB00] dealed with N-gram technique for detection. The authors collected 149,882 malicious files and 4934 benign files and divided them into train and test set. The authors tried to find the optimal size of n-gram, the number of nearest neighbors for the knn algorithm and the limit that marks the nature of the unknown instance as malware. We can see the results (detection ratio and False Positive ratio) in the diagram Figure 12. Using the parameters: n-gram size is 6, the number of nearest neighbors for the knn algorithm is 7 and the limit that marks the nature of the unknown

instance as malware is 2, provided the optimal combination between the maximum detection ratio and lowest False Positive ratio.

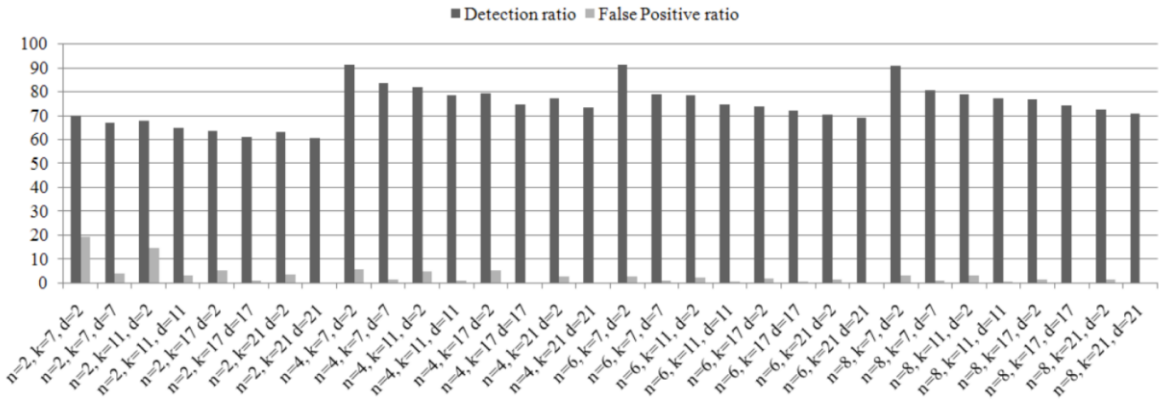


Figure 12. Detection and false positive rate

In the article [CNRE16] Cohen, Nissim, Rokach and Elovici used a structural feature extraction methodology for the detection malicious office files. Every file was unzipped and every sub file was entered as a path in the file. Thus, every file was a unique list with different paths. By taking a big database (16,108 benign and 830 malicious .docx files), they ran four experiments. The first one includes using machine learning with different classifiers (J48, Random forest, Logistic regression etc.) in a 10-fold cross-validation. The second experiment includes checking the paths without numbers. The third experiment includes compression using N-gram and the fourth includes comparison of the detection rate with Anti-Virus softwares. We can see some the results of couple of those experiments. Figure 11 presents the comparison between the different AUC of the classifiers (from the top of 200 on, Random forest of 500 trees shows the better performances). Figure 14 presents the True-Positive Ratio of SFEM methods relatively to another Anti-Virus software (SFEM presented the best ratio of 97% TP):

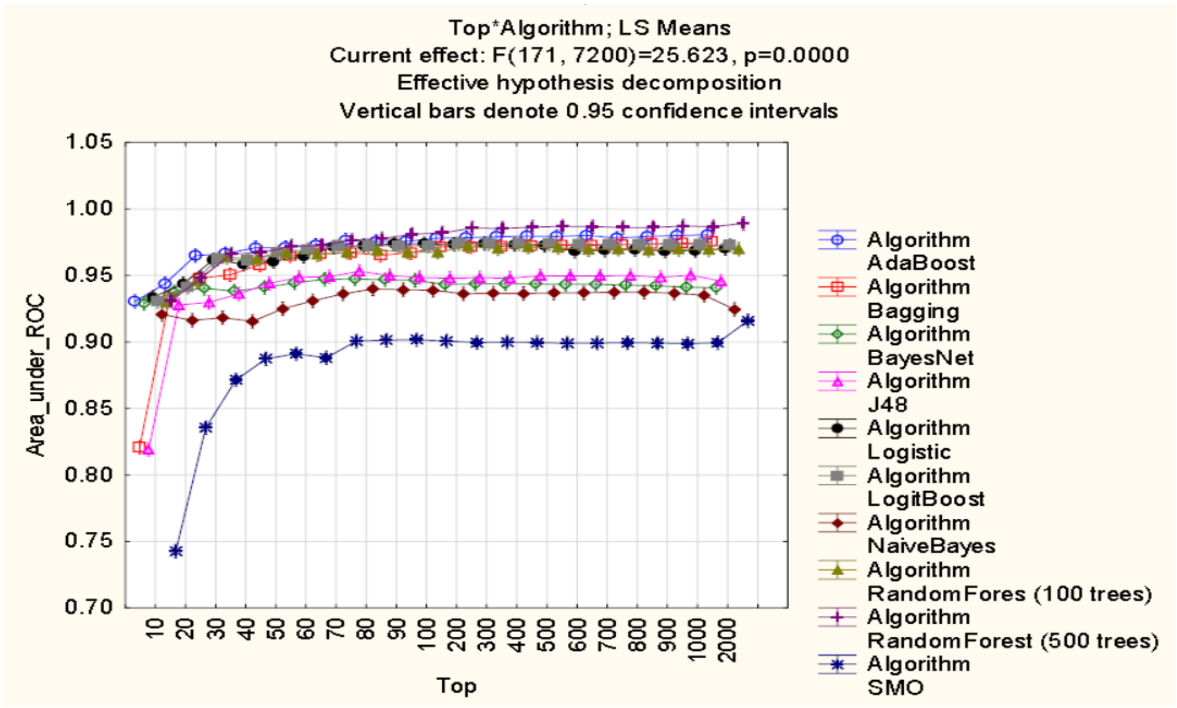


Figure 13.Comparasion of the average AUC of the selected classifiers on different top-features datasets

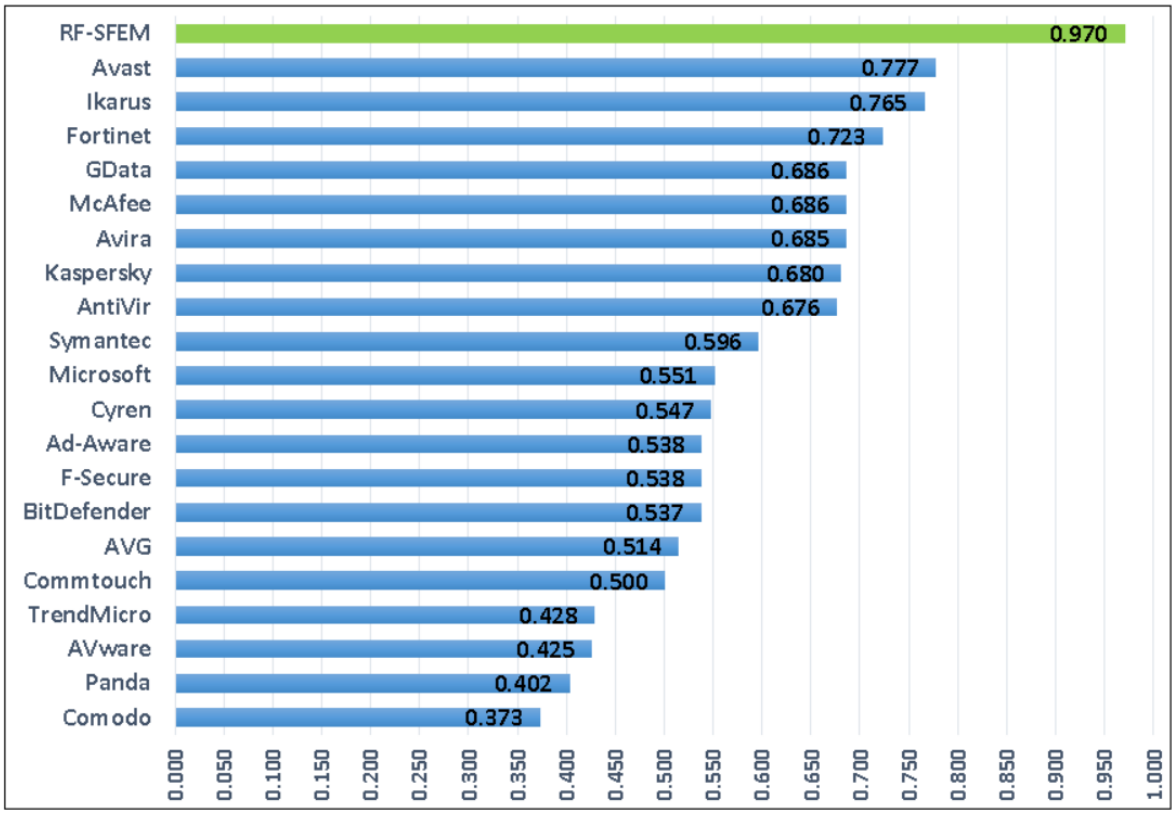


Figure 14.The TPR of RF-SFEM method compared to anti-viruses commonly used by organizations

The results showed that using machine learning improve the detection rate in very high percentages. By using the random forest classifier, the authors achieved an AUC of 99.12%, True positive rate of 97% and false positive rate of 4.9%.

Nissim, Cohen, Glezer and Elovici [NCGE14] displayed how to implement the same machine learning methods from the office file to PDF. The research used Feature extraction on the PDF structure. Creation of paths from top to bottom and every path can be a feature.

Microsoft RTF specification 1.9.1 [Serv08], for thorough understanding of the file structure. The file consists every details and examples of every Group, Control word etc. Also, the document explains the different between the header and the documents group.

The book "JPEG still image data compression standard" [PeMi93] for thorough understanding of the file structure. In the book we found out every detail about the way JPEG compression is working, the vary markers.

V. METHODS

In our project, we are detecting malicious files with machine learning classifiers trained on datasets containing features generated by the feature extraction methods that we will present. Feature extraction is a method that takes data and divides it to parts (so called features). The division depends on the data and the way it's built. (In Sections "Background" and "RTF File Structure" we elaborated about the patterns in RTF file).

In this section, we will present two different methods for feature extraction. The first method is a predefined, knowledge-based, finite set of features, based on known characteristics of malicious and benign files. The second method is feature extraction methodology based on the structure of RTF file presented earlier. In addition we will elaborate on the feature selection methods and feature representation method. We will also present the data collection we used for evaluation and describe the process of dataset creation.

To examine our methods, we collected both malicious and benign RTF files from *VirusTotal*. Our collection is composed of 1,224 malicious and 2,204 benign files. To sum it up, the data collection consists 3,428 RTF files. The percentages are approximately 0.36 malicious and approximately 0.64 benign files.

A. *Dataset Creation – Structural*

Visually, you can find the whole process of the feature extraction in Figure 15.

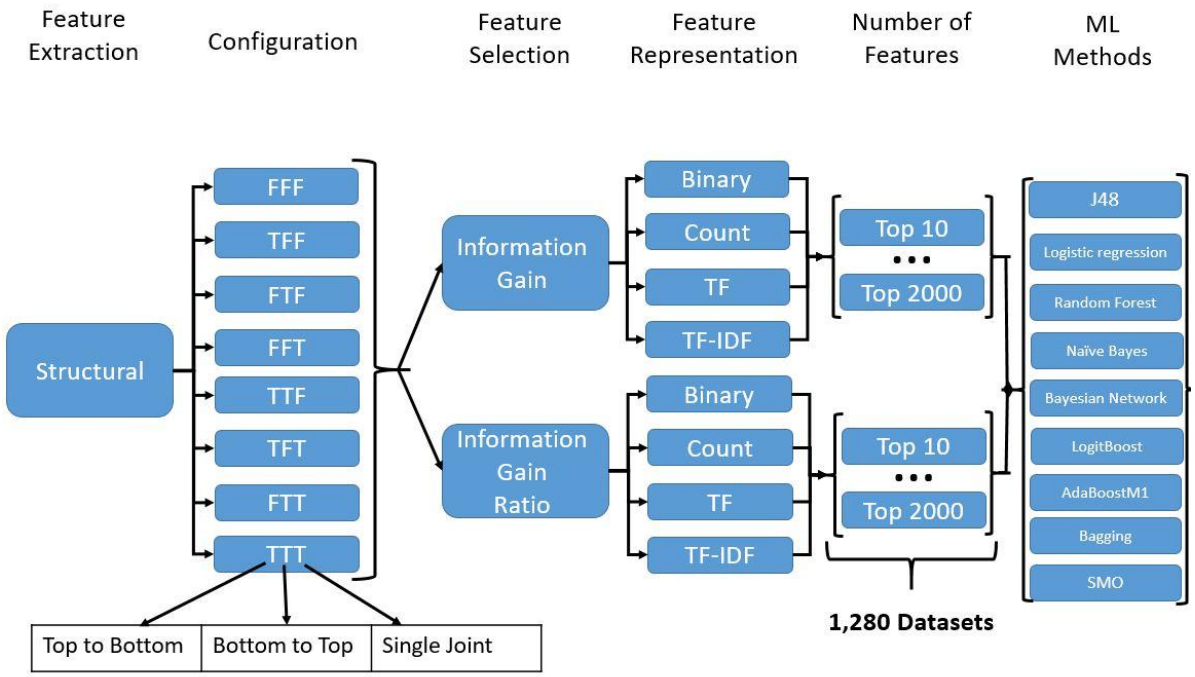


Figure 15. The Whole Process of Feature Extraction

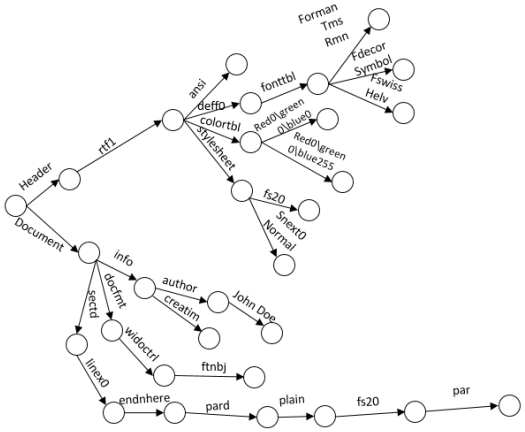
1. *Feature Extraction*

The purpose of feature extraction process is to extract discriminative features from the examined element. The extracted features will be leveraged using machine learning algorithms for the detection of the correct class of the element.

As we mentioned in the start of the section, the partition to features depends on the data and the way it's built. We found out that RTF files have a hierarchical structure of groups which contain control words (Figure 16), thus we decided to use this fact to create a strings of features. Every control word at the beginning of a group starts a feature string (top down configuration) and also, it is a feature by itself (single joint configuration). Every following control word is added to the feature string and also becomes a feature by itself. In this way, when the parser finishes extracting the features from a RTF file, we get every control word as a feature and features that are constructed by series of control words. For example: /rtf , /rtf/ansi, ansi, /rtf/ansi/deff0 and deff0 are all features. Moreover, we decided to create another set of features that are constructed by traversing the hierarchical structure of the file from bottom to top (bottom to top configuration). Thus, we collected all the information and unique parts of RTF file we could get to indicate if the file is malicious. As part of the research, for each configuration of the eight extraction configuration (Figure 15) we created a dataset of unique features collected from all the RTF files and their number of occurrences in each file in our collection.

```
{\rtf1\ansi\deff0{\fonttbl{\f0\froman Tms Rmn;}{\f1\fdecor Symbol;}{\f2\fswiss Helv;}}
{\colortbl;\red0\green0\blue0;\red0\green0\blue255;\red0\green255\blue255;\red0\green255\blue0;\red25
5\green0\blue255;\red255\green0\blue0;\red255\green0\blue255;\red255\green255\blue0;\red255\green255\blue255;}
{\stylesheet{\fs20 \snext0 Normal;}}{\info{\author John Doe}{\creatim\yr1990\mo7\dy30\hr10\min48}
{\version1}{\edmins0}{\nofpages1}{\nofwords0}{\nofchars0}{\vern8351}}
\widoctrl\ftnbj \sectd\linex0\endnhere \pard\plain \fs20 This is plain text.\par}
```

Example Source: Microsoft RTF Specification 1.9.1



Structural Path:
Root\Header\rtf1
rtf1
Root\Header\rtf1\ansi
ansi
Root\Header\rtf1\deff0
deff0
Root\Header\rtf1\deff0\Forman Tms Rmn
Forman Tms Rmn
…
Root\Document\info

Figure 16. Structural path of RTF

2. *Feature Selection*

There are three main approaches for feature selection: *filter methods*, *wrapper methods,* and *embedded methods*. *Filter methods* use a measure to evaluate the correlation of each individual feature to the class (malicious/benign). After applying a filter on a dataset, each feature gets a rank which quantifies its expected contribution in the classification task, from which the features with the top X ranks are used to train machine learning algorithms. *Wrapper methods* conduct a search for a good subset of features using the learning algorithm itself as part of the evaluation function. *Embedded methods* perform feature selection as part of the learning procedure and are usually specific to given learning algorithms (i.e., classification trees, etc.).

Since the total number of unique features extracted from our collection is very large, a filter method best fits our needs for a number of reasons: it allows us to select the top X prominent features; it scales easily to high dimensional datasets; and it is computationally simple, fast, and independent of the classification algorithm, thus, enabling us to compare the performances of the different classification algorithms on the same subset of features. We rejected *wrapper methods* for two reasons. This first is because they depend on a classifier in order to select a good subset of features, and each classifier would select a different subset. The second reason is that the process of searching a subset of features in a very high dimensional dataset such as ours is too heavy. We rejected *embedded methods,* since this approach depends on the classifier.

Although we choose the *filter method*, we are aware of the disadvantages. The first disadvantage is that with this approach, the potential contribution from the interaction between the features is neglected. For example, when brought together, a subset of features might contribute significantly to the classification task. However, individual features within the subset might obtain a low rank, and thus may not be included in the top X features. The second disadvantage is that when using the top X features ranked by a specific filter, one might obtain sets of features that contribute equally to the

discrimination between classes. However, it is more efficient to use only a single feature from each set and thereby accommodate other features with different contributions.

In this phase we used two methods. The first is information gain (IG) which evaluates the worth of an attribute by measuring how the attributes contributes to discriminate between malicious and benign class. The second is information gain ratio (GR). GR is an extension of IG, which normalizes the value of IG using the so called split info. We used the two methods of selection that give a rank to each feature and sort them according to their prominence to choose the top X ranked features.

3. *Feature Representation*

Feature representation determines how the presence (and importance) of a feature in a file is represented. We used four feature representation methods: Binary, Count, TF and TF-IDF. Binary representation is used to indicate the presence (1) or absence (0) of a feature within the file. TF shows the term frequency in the file normalized by dividing its frequency by the most frequent term in the document. TFIDF is a weighting method often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a term is to a document in a collection. It combines the frequency of a term in the document (TF) and its frequency in the document collection, denoted by DF. The definition of TFIDF is multiplying the term frequency (TF) by the $\mathrm{IDF} = \log_2(\mathrm{N/DF})$, where N is the number of documents in the entire collection and DF is the number of files in which the term appears. In our case a term is a feature extracted from a RTF file.

4. *Number of Top Selected Features*

In order to determine how the number of top selected features affects the detection rates, we also took into consideration the top subsets of the most prominent features. We created twenty datasets using the 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1,000, and 2,000 most prominent features selected by a feature selection method.

At the end of the process, we hold 20 datasets (Top 10, 20 … 1000, 2000) for each representation method (binary, count, TF and TFIDF), for each two feature selection method (information gain and information gain ratio) and for each of eight feature extraction configuration method (FFF, TFF, FTF, FFT, TTF, TFT, FTT and TTT). To sum up, we hold 1,280 datasets. We applied several machine learning classifiers on each of these datasets.

B. Dataset Creation – Knowledge-Based

1. *Feature Extraction*

The purpose of feature extraction process is to extract discriminative features from the examined element. The extracted features will be leveraged using machine learning algorithms for the detection of the correct class of the element.

The creation of knowledge-based features demanded thorough understanding of former attacks in which RTF files were used, exploited vulnerabilities in RTF files and the structure of RTF file (the role of each control word, hierarchy of the file etc.). Table 11 presents the list of features we implemented for the RTF file, including the feature's name, the element of the file that the feature is associated with, the description, feature type (simple, attack-based or expert), data type (binary, integer, etc.).

ID	Feature Name	Element	Description	Feature Type	Data Type
1	File_size	File	Checking the RTF size and detect suspicious values (0 kb)	Simple	Integer
2	File_extension	File	Checking if it's a RTF file (starting {\rtf)	Attack-based	Integer
3	Group_structure_incorrect_num	Group	Number of groups don't have open and close brackets.	Simple	Integer
4	CW_rtf_incorrect_val	Control Word	Checking the /rtf has the digit 1 after it (version number)	Attack based	Binary
5	CW_ole_content_url_num	Control Word	Number of OLE objects contains URL	Expert	Integer
6	CW_List_Incorrect_val_num	Control Word	Number of incorrect List values. List value must be between 1 to 9	Attack-based	Integer
7	CW_pict_content_url_num	Control Word	Number of pict contains URL	Attack-based	Integer
8	CW_length_Incorrect_num	Control Word	Number of control words that their length is more than 32 characters	Attack-based	Integer
9	CW_fromhtml_incorrect_val_num	Control Word	Number of fromhtml control word value other than 1 (version number). The correct control word: \fromhtml1.	Expert	Integer
10	CW_fprq_incorrect_val_num	Control Word	Number of fprq control word value other than 0, 1 or 2.	Expert	Integer
11	CW_fbias_incorrect_val_num	Control Word	Number of fbias control word value other than 0 or 1.	Expert	Integer
12	CW_fcharset_incorrect_val_num	Control Word	Number of fcharset control word value as number longer than 5 digits or negative	Expert	Integer
13	CW_deflang_incorrect_val_num	Control Word	Number of deflang control word value as number longer than 5 digits or negative	Expert	Integer

14	CW_fid_incorrect_val_num	Control Word	Number of fid control word value as negative number	Expert	Integer
15	CW_frealative_incorrect_val_num	Control Word	Number of frealative control word value as negative number	Expert	Integer
16	CW_red_incorrect_val_num	Control Word	Number of red control word value as negative number or more than 255	Expert	Integer
17	CW_green_incorrect_val_num	Control Word	Number of green control word value as negative number or more than 255	Expert	Integer
18	CW_blue_incorrect_val_num	Control Word	Number of blue control word value as negative number or more than 255	Expert	Integer
19	CW_ctint_incorrect_val_num	Control Word	Number of ctint control word value as negative number or more than 255	Expert	Integer
20	CW_cshade_incorrect_val_num	Control Word	Number of cshade control word value as negative number or more than 255	Expert	Integer
21	CW_s_incorrect_val_num	Control Word	Number of s control word value as negative number or more than 65535	Expert	Integer
22	CW_lsdlockeddef_incorrect_val_num	Control Word	Number of lsdlockeddef control word value other than 0 or 1.	Expert	Integer
23	CW_listid_incorrect_val_num	Control Word	Number of listid control word value between -1 to -5.	Expert	Integer
24	CW_listsimple_incorrect_val_num	Control Word	Number of listsimple control word value other than 0 or 1.	Expert	Integer
25	CW_listrestarthdn_incorrect_val_num	Control Word	Number of listrestarthdn control word value other than 0 or 1.	Expert	Integer
26	CW_levelnfc_incorrect_val_num	Control Word	Number of levelnfc control word incorrect value. Correct value is between 0-7, 10-68 or 255	Expert	Integer
27	CW_leveljcn_incorrect_val_num	Control Word	Number of leveljcn control word value other than 0, 1 or 2.	Expert	Integer
28	CW_levelold_incorrect_val_num	Control Word	Number of levelold control word value other than 0 or 1.	Expert	Integer
29	CW_levelfollow_incorrect_val_num	Control Word	Number of levelfollow control word value other than 0, 1 or 2.	Expert	Integer
30	CW_levellegal_incorrect_val_num	Control Word	Number of levellegal control word value other than 0 or 1.	Expert	Integer
31	CW_proptype_incorrect_val_num	Control Word	Number of levelfollow control word value other than 3, 5, 11, 30 or 64.	Expert	Integer
32	CW_yr_incorrect_val_num	Control Word	Number of incorrect yr values. yr value must be between 1900 to 2017	Expert	Integer
33	CW_mo_incorrect_val_num	Control Word	Number of incorrect mo values. mo value must be between 1 to 12	Expert	Integer
34	CW_dy_incorrect_val_num	Control Word	Number of incorrect dy values. dy value must be between 1 to 31	Expert	Integer
35	CW_hr_incorrect_val_num	Control Word	Number of incorrect List values. hr value must be between 0 to 23	Expert	Integer
36	CW_min_incorrect_val_num	Control Word	Number of incorrect min values. min value must be between 0 to 59	Expert	Integer

37	CW_sec_incorrect_val_num	Control Word	Number of incorrect sec values. sec value must be between 0 to 59	Expert	Integer
38	CW_doctype_incorrect_val_num	Control Word	Number of doctype control word value other than 0, 1 or 2.	Expert	Integer
39	CW_stylesortmethod_incorrect_val_num	Control Word	Number of stylesortmethod control word value other than 0, 1, 2, 3 or 4.	Expert	Integer
40	CW_viewkind_incorrect_val_num	Control Word	Number of viewkind control word value other than 0, 1, 2, 3, 4 or 5.	Expert	Integer
41	CW_viewscale_incorrect_val_num	Control Word	Number of incorrect viewscale values. viewscale value must be between 0 to 100.	Expert	Integer
42	CW_viewzk_incorrect_val_num	Control Word	Number of viewzk control word value other than 0, 1, 2 or 3.	Expert	Integer
43	CW_viewbksp_incorrect_val_num	Control Word	Number of viewbksp control word value other than 0 or 1.	Expert	Integer
44	CW_fet_incorrect_val_num	Control Word	Number of fet control word value other than 0, 1 or 2.	Expert	Integer
45	CW_revprop_incorrect_val_num	Control Word	Number of revprop control word value other than 0, 1, 2, 3 or 4.	Expert	Integer
46	CW_revbar_incorrect_val_num	Control Word	Number of revbar control word value other than 0, 1, 2 or 3.	Expert	Integer
47	CW_protlevel_incorrect_val_num	Control Word	Number of protlevel control word value other than 0, 1, 2 or 3.	Expert	Integer
48	CW_stexflow_incorrect_val_num	Control Word	Number of stexflow control word value other than 0, 1, 2, 3, 4 or 5.	Expert	Integer
49	CW_horzvert_incorrect_val_num	Control Word	Number of horzvert control word value other than 0 or 1.	Expert	Integer
50	CW_twoinone_incorrect_val_num	Control Word	Number of twoinone control word value other than 0, 1, 2, 3 or 4.	Expert	Integer
51	CW_fftypetxt_incorrect_val_num	Control Word	Number of fftypetxt control word value other than 0, 1, 2, 3, 4 or 5.	Expert	Integer
52	CW_ffrecalc_incorrect_val_num	Control Word	Number of ffrecalc control word value other than 0 or 1.	Expert	Integer
53	CW_ffhaslistbox_incorrect_val_num	Control Word	Number of ffhaslistbox control word value other than 0 or 1.	Expert	Integer
54	CW_ffsize_incorrect_val_num	Control Word	Number of ffsize control word value other than 0 or 1.	Expert	Integer
55	CW_ffprot_incorrect_val_num	Control Word	Number of ffprot control word value other than 0 or 1.	Expert	Integer
56	CW_ffownstat_incorrect_val_num	Control Word	Number of ffownstat control word value other than 0 or 1.	Expert	Integer
57	CW_ffownhelp_incorrect_val_num	Control Word	Number of ffownhelp control word value other than 0 or 1.	Expert	Integer
58	CW_fftype_incorrect_val_num	Control Word	Number of fftype control word value other than 0, 1 or 2.	Expert	Integer
59	CW_sbauto_incorrect_val_num	Control Word	Number of sbauto control word value other than 0 or 1.	Expert	Integer

60	CW_saauto_incorrect_val_num	Control Word	Number of saauto control word value other than 0 or 1.	Expert	Integer
61	CW_slmult_incorrect_val_num	Control Word	Number of slmult control word value other than 0 or 1.	Expert	Integer
62	CW_pnlvl_incorrect_val_num	Control Word	Number of incorrect pnlvl values. pnlvl value must be between 1 to 9	Expert	Integer
63	CW_abslock_incorrect_val_num	Control Word	Number of abslock control word value other than 0 or 1.	Expert	Integer
64	CW_dropcapli_incorrect_val_num	Control Word	Number of incorrect dropcapli values. dropcapli value must be between 1 to 10	Expert	Integer
65	CW_dropcapt_incorrect_val_num	Control Word	Number of dropcapt control word value other than 1 or 2.	Expert	Integer
66	CW_absnoovrlp_incorrect_val_num	Control Word	Number of absnoovrlp control word value other than 0 or 1.	Expert	Integer
67	CW_trautofit_incorrect_val_num	Control Word	Number of trautofit control word value other than 0 or 1.	Expert	Integer
68	CW_shpfblwtxt_incorrect_val_num	Control Word	Number of shpfblwtxt control word value other than 0 or 1.	Expert	Integer
69	CW_shpwr_incorrect_val_num	Control Word	Number of shpwr control word value other than 0, 1, 2, 3, 4 or 5.	Expert	Integer
70	CW_shpwrk_incorrect_val_num	Control Word	Number of shpwrk control word value other than 0, 1, 2 or 3.	Expert	Integer
71	CW_dpfillpat_incorrect_val_num	Control Word	Number of incorrect dpfillpat values. dpfillpat value must be between 0 to 25	Expert	Integer
72	CW_dpastartw_incorrect_val_num	Control Word	Number of dpastartw control word value other than 0, 1, 2 or 3.	Expert	Integer
73	CW_dpastartl_incorrect_val_num	Control Word	Number of dpastartl control word value other than 0, 1, 2 or 3.	Expert	Integer
74	CW_dpaendw_incorrect_val_num	Control Word	Number of dpaendw control word value other than 0, 1, 2 or 3.	Expert	Integer
75	CW_dpaendl_incorrect_val_num	Control Word	Number of dpaendl control word value other than 0, 1, 2 or 3.	Expert	Integer
76	CW_wbitmap_incorrect_val_num	Control Word	Number of incorrect dpfillpat values. dpfillpat value must be between 1 to 8	Expert	Integer
77	CW_tblindtype_incorrect_val_num	Control Word	Number of tblindtype control word value other than 0, 1, 2 or 3.	Expert	Integer
78	CW_mcGpRule_incorrect_val_num	Control Word	Number of mcGpRule control word value other than 0, 1, 2, 3 or 4.	Expert	Integer
79	CW_mscr_incorrect_val_num	Control Word	Number of mscr control word value other than 0, 1, 2, 3, 4 or 5.	Expert	Integer
80	CW_msty_incorrect_val_num	Control Word	Number of msty control word value other than 0, 1, 2 or 3.	Expert	Integer
81	CW_animtext_incorrect_val_num	Control Word	Number of incorrect animtext values. animtext value must be between 0 to 8	Expert	Integer
82	CW_lbr_incorrect_val_num	Control Word	Number of lbr control word value other than 0, 1, 2 or 3.	Expert	Integer

83	CW_hres_incorrect_val_num	Control Word	Number of incorrect hres values. hres value must be between 0 to 6	Expert	Integer
84	CW_fbidis_num	Control Word	Total amount of fbidis control word in the file	Simple	Integer
85	CW_filetbl_num	Control Word	Total amount of filetbl control word in the file	Simple	Integer
86	CW_colortbl_num	Control Word	Total amount of colortbl control word in the file	Simple	Integer
87	CW_stylesheet_num	Control Word	Total amount of stylesheet control word in the file	Simple	Integer
88	CW_listtable_num	Control Word	Total amount of listtable control word in the file	Simple	Integer
89	CW_revtbl_num	Control Word	Total amount of revtbl control word in the file	Simple	Integer
90	CW_rsidtbl_num	Control Word	Total amount of rsidtbl control word in the file	Simple	Integer
91	CW_mmathpr_num	Control Word	Total amount of mmathPr control word in the file	Simple	Integer
92	CW_generator_num	Control Word	Total amount of generator control word in the file	Simple	Integer
93	CW_info_num	Control Word	Total amount of info control word in the file	Simple	Integer
94	CW_fonttbl_num	Control Word	Total amount of fonttbl control word in the file	Simple	Integer
95	CW_xmlnstbl_num	Control Word	Total amount of xmlnstbl control word in the file	Simple	Integer
96	CW_sect_num	Control Word	Total amount of sect control word in the file	Simple	Integer
97	Group_empty_num	Group	Number of empty groups in the file	Expert	Integer
98	CW_bin_num	Control Word	Total amount of bin control word in the file	Simple	Integer

2. *Feature Selection*

The features we constructed for the Knowledge-Based method are based on the understanding of former attacks and how can one exploit the RTF file specification for construction of a malicious file; therefore our features are very general and are not based on a specific case of a malicious RTF file and it is safe to say that using all the features without a feature selection process is safe and will not cause overfitting.

3. *Feature Representation*

In contrast to the Structural feature extraction method, we used only the count method (Sums the number of appearances of a feature in the file) for representation of the different features.

At the end of the process, we hold one dataset, containing 97 features and the number of instances of each feature in each file in our collection. We applied several machine learning classifiers on the dataset for evaluation of our proposed method for feature extraction.

C. Machine Learning Algorithms

We employed nine commonly used machine learning classification algorithms on the datasets created. We chose the following classifiers: *J48, Random Forest* (RF), *Naïve Bayes* (NB), *Bayesian Network* (BN), *Logistic Regression* (LR), *LogitBoost* (LB), *Sequential Minimal Optimization (SMO), Bagging* and *AdaBoost* (AB). Every different method is representing different classifier category: classification trees (*J48, RF)*, Bayesian classifiers *(NB, BN)*, function based classifiers *(LR, SMO)*, and meta classifiers *(LB, Bagging, AB)*.

We applied all of the above mentioned machine learning classifiers using the *Weka* data mining software with *WEKA's* default configuration. We used the following configurations of classifiers: *J48* with a confidence factor (used for pruning) of 0.25 and a minimum of two instances per leaf; *Random Forest* classifier twice, once with 100 trees and again with 500 trees, with no limitation regarding the depth of the trees; *LogitBoost* classifier based on the *DecisionStump* classifier; *SMO* classifier with a polynomial kernel; *Bagging* classifier based on the *REPTree* classifier; and *AdaBoost* based on the *DecisionStump* classifier.

VI. EVALUATION

A. Research Questions

In order to evaluate the effectiveness of the proposed methods presented earlier (structural feature extraction methodology and knowledge-based features) for the detection of unknown malicious *.rtf documents using machine learning algorithms, we first wanted to determine which configuration yields the best detection accuracy measures. The research questions are as follows:

1. *Structural Feature Extraction Method*

1. Are machine learning algorithms trained on features extracted using the structural feature extraction methodology presented earlier useful for the efficient detection of malicious RTF documents?
2. Which feature extraction configuration provides the best detection results?
3. Which feature selection method provides the best detection results: *Information Gain* or *Information Gain Ratio?*
4. Which feature representation method provides the best detection results: *Binary*, *Count*, *TF* or *TFIDF?*
5. Which top-features dataset provides the best detection results: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1,000, or 2,000?
6. Which classifier provides the best detection results: *Naïve Bayes*, *Bayes Network*, *J48*, *Random Forest*, *Logistic Regression*, *LogitBoost*, *SMO*, *Bagging*, or *AdaBoost?*
7. Which configuration of feature extraction, feature selection, feature representation, top-feature selection, and classifier provides the best detection results?

2. *Knowledge-Based Feature Extraction Method*

1. Are machine learning algorithms trained on the knowledge-based features presented earlier useful for the efficient detection of malicious RTF documents?
2. Which classifier provides the best detection results: *Naïve Bayes*, *Bayes Network*, *J48*, *Random Forest*, *Logistic Regression*, *LogitBoost*, *SMO*, *Bagging*, or *AdaBoost?*

3. *Both Structural and Knowledge-Based Feature extraction Methods*

1. Does the best configuration of the structural feature extraction and Knowledge based feature extraction methods provide better detection results than existing anti-virus engines?

B. Evaluation Measures

For evaluation purposes, we measured the AUC (area under the receiver operating characteristic (ROC) curve). The ROC curve is created by plotting the TPR (true positive rate) against the FPR (false positive rate) at various threshold settings. In order to achieve high AUC, high TPR and low FPR are needed to check at each threshold. The AUC is a preferred measure (over the accuracy measure) for comparing machine learning algorithms applied on datasets (balanced or imbalanced binary or multiclass) as suggested by [HuLi05].

C. Experimental Design

To answering the Research questions that we wrote above, we designed three Experiments. Every experiment will help us to collect the relevant information for the research questions.

1. Experiment 1 – Finding the Best Overall Configuration for Structural

The first experiment relates to research question 1-7 in the Structural Feature Extraction Method category. The experiment will evaluate the results achieved by applying each classifier of the nine on 1280 datasets and compare the results between them. The results will be related to the average AUC achieved with all the different configurations (feature extraction, feature selection, feature representation, top-feature selection and classifier).

2. Experiment 2 – Detection results of knowledge-based

The second experiment relates to research question 1-2 in the Knowledge-Based Feature Extraction Method category. The experiment will evaluate the results achieved by applying each classifier of the nine on the dataset and compare the AUC results between them.

3. Experiment 3 – Comparison with Anti-Virus Engines

The third experiment relates to the research question in the Both Structural and Knowledge-Based feature extraction Methods category. The experiment will compare the TPR provided by the best configuration of the Structural and Knowledge-Based feature extraction methods with the different Anti-Virus softwares. We used VirusTotal's online web service which provided analysis of 64 commonly used Anti-Virus engines, to analyze the malicious files in our collection. We analyzed the report produced by VirusTotal to compute the TPR for each Anti-Virus engine.

D. Results

1. Results of Experiment 1

Experiment number 1 examined the performance of the structural feature extraction method. It answers the seven research questions on the Structural Feature Extraction Method category.

The first research question checks if the structural feature extraction method is a good way of detection malicious files. The answer for this question is yes and we will demonstrate it through the other questions.

We examined the five parts of the process (Extraction, Selection, Representation, Top features and Classifier) independently and together. In Figure 17 we can observe the average AUC of every feature extraction configuration. The best configuration is FFT which provided 0.962 average AUC. From those results, we can say that the best configuration is the Single Joint. In addition, we can see that the next configuration with the second best AUC is TFT (0.9615), which confirms the importance of the Single Joint.

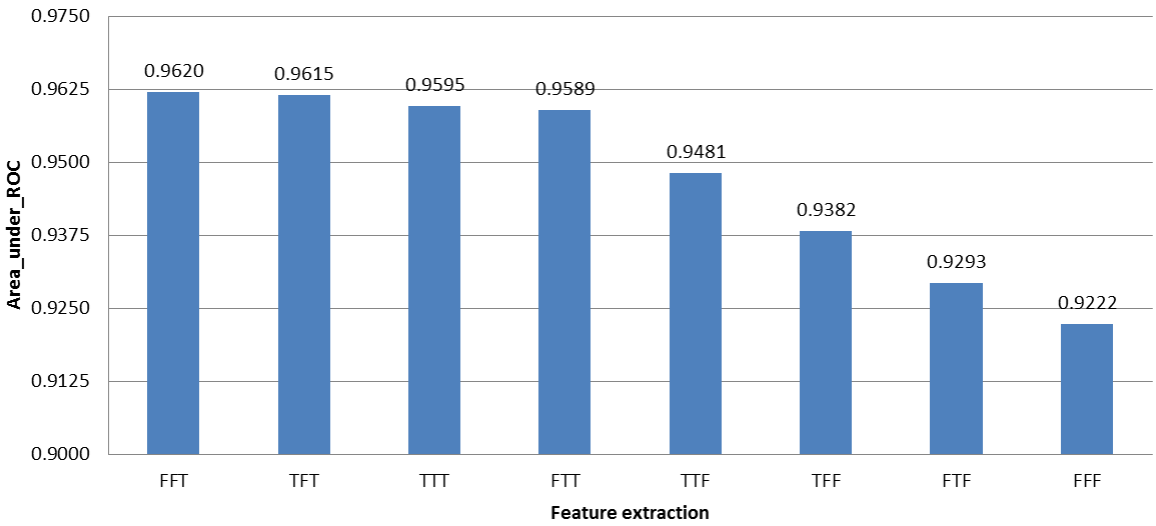


Figure 17. Comparison between the different configurations – Structural Feature Extraction method

In Figure 18 we can observe the average AUC of every feature selection method. The best feature selection method is Information Gain which provided 0.9731 average AUC.

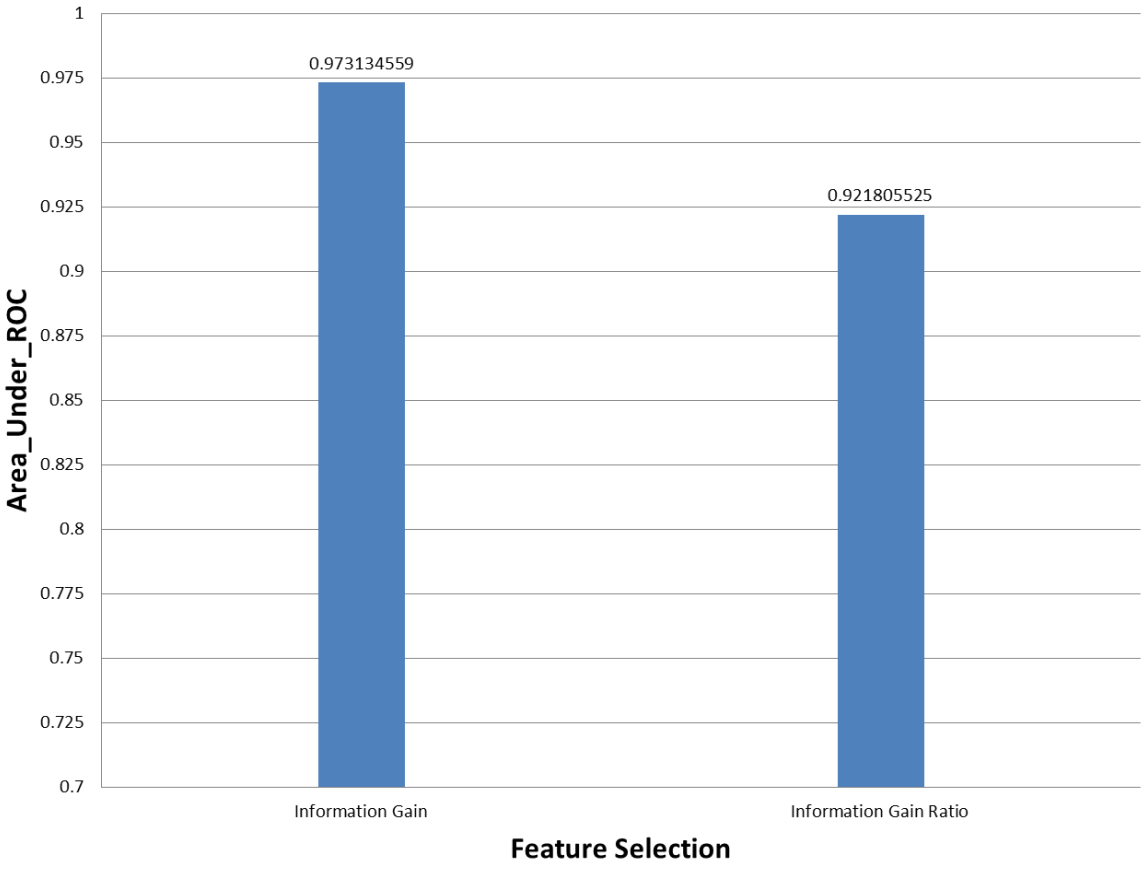


Figure 18. Comparison between the different selections – Structural Feature Extraction method

In Figure 19 we can observe the average AUC of every representation method. The best representation method is Binary which provided 0.9634 average AUC.

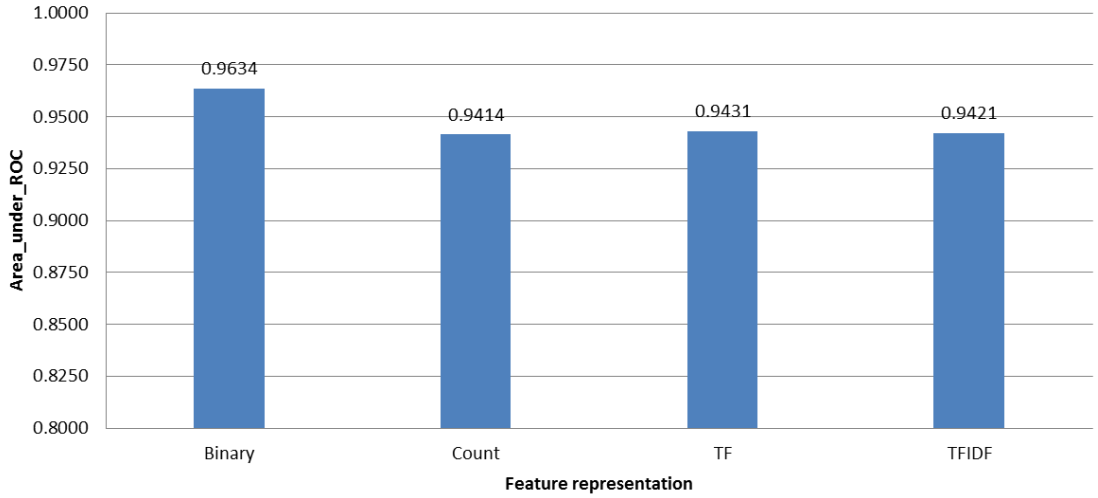


Figure 19. Comparison between the different representations – Structural Feature Extraction method

In Figure 20 we can observe the average AUC of every top features. The best top features is 1000.

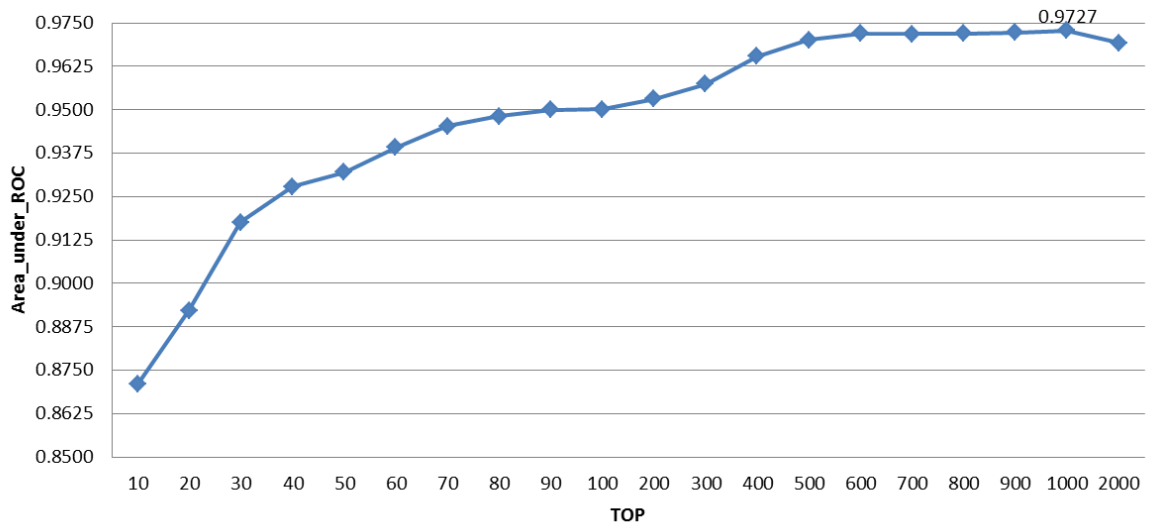


Figure 20. Comparison between the different Top Features – Structural Feature Extraction method

In Figure 21 we can observe the average AUC of every classifier. The best classifier is Random Forest.

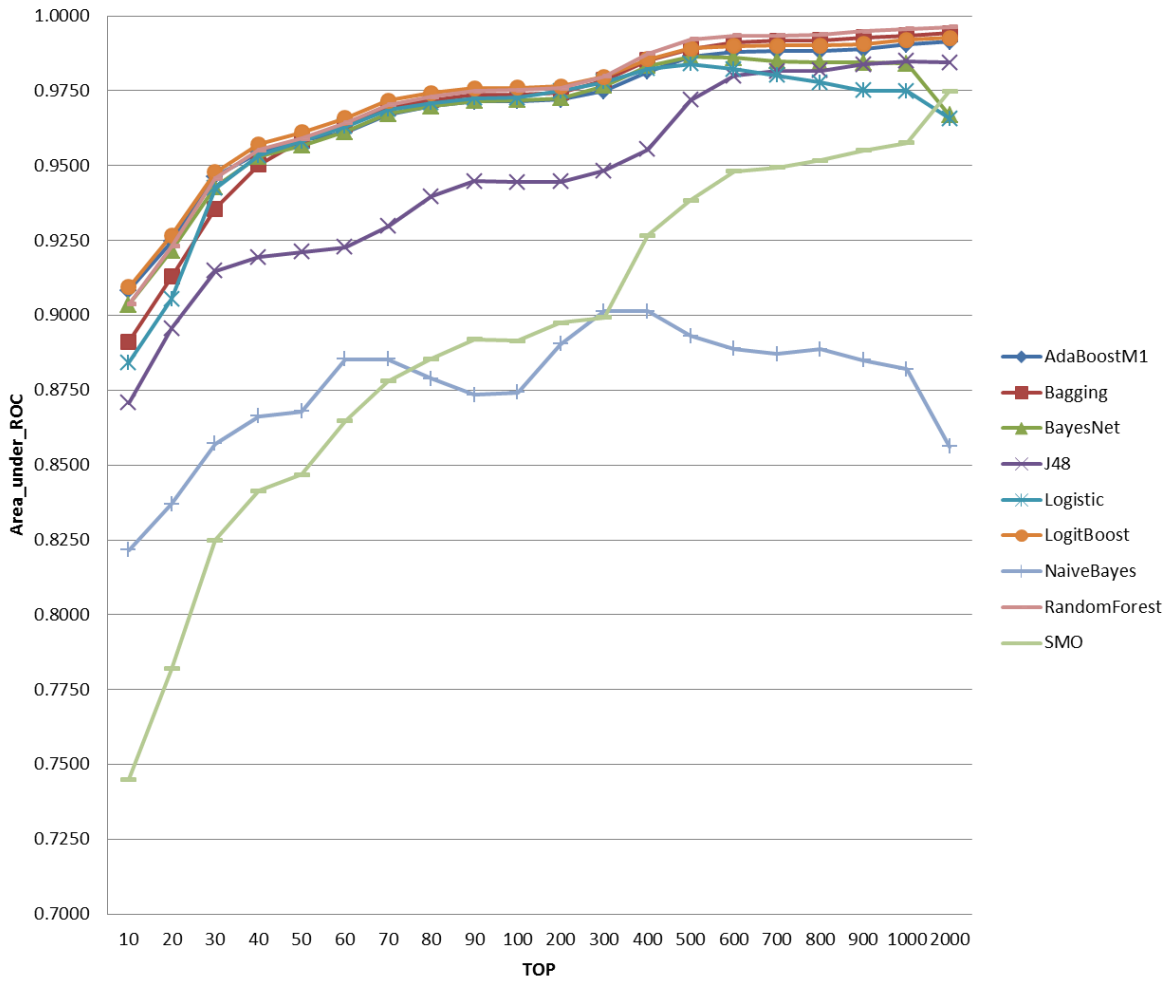


Figure 21. Comparison between the different classifiers – Structural Feature Extraction method

To answer the sixth research question, determining the best configuration, we examined the configuration with the highest AUC. We searched for the most prevalent condition for each factor. In Table 4 we can observe TPR, FPR, AUC and F-Measure of the best combination (Configuration, Selection, Representation, Top features and Classifier). The best combination is TFT extraction, TF representation, Info Gain Ratio selection, 800 Top features and Random Forest classifier which provided 0.9787 TPR, 0.0049 FPR and AUC 0.9985. It's important to mention that this is the combination that provided average best performances in a 10-fold cross-validation format. In conclusion, the results provided by machine learning algorithms trained on the structural feature extraction method are useful for efficient detection of malicious RTF files.

Feature Ext.	Feature Rep.	Feature Select.	Top Features	Classifier	TPR	FPR	AUC	F-Measure
TFT	TF	Info Gain Ratio	800	Random Forest	0.9787	0.0049	0.9985	0.9847

Table 4. TPR, FPR, AUC and F-Measure of the best combination (Configuration, Selection, Representation, Top features

2. *Results of Experiment 2*

Experiment number 2 examined the performance of the knowledge-based feature extraction method. It answers the two research question on the Knowledge-Based Feature Extraction Method category. In Figure 22 we can observe the average accuracy ratio of every classifier. The best classifier is Random Forest which provided 0.9943 average AUC. Those results indicate that Knowledge-Based feature extraction method is a good detection method which provides high AUC values.

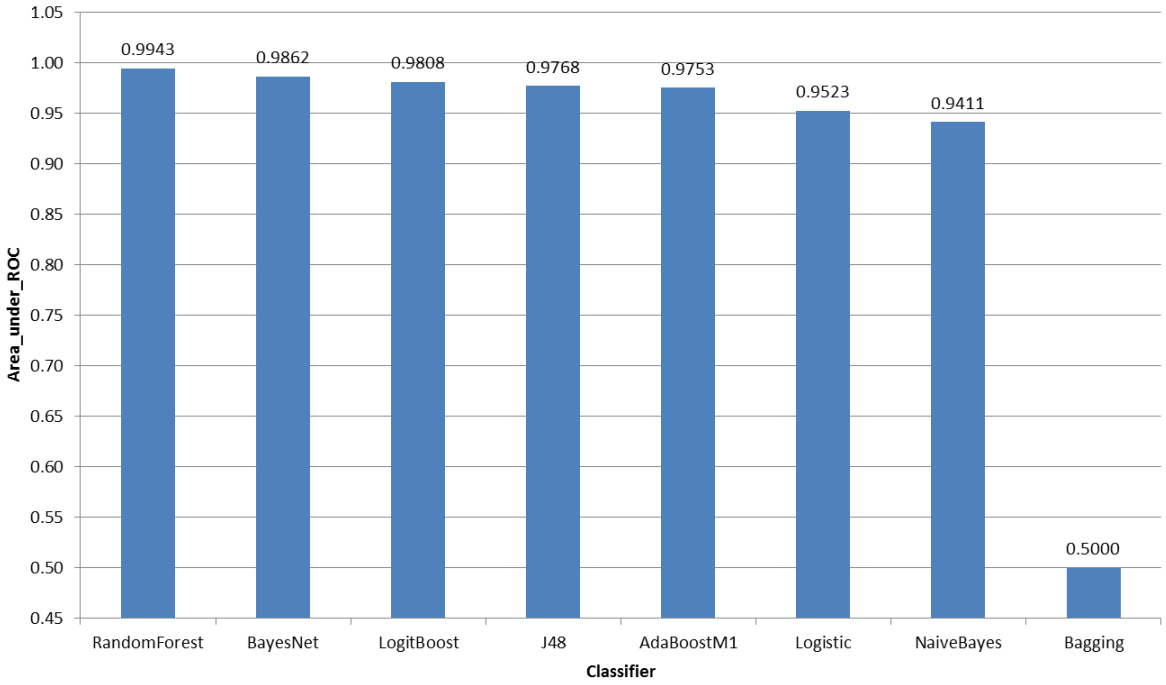


Figure 22. Knowledge-Based Feature Extraction method AUC

3. *Results of Experiment 3*

Experiment number 3 compares the true positive rate (TPR) between the best configurations found in the previous experiment of the structural feature extraction method, knowledge-based feature extraction method with the Anti-Virus softwares detection TPR. This will show the best way for detection malicious RTF files. In Figure 23 we can observe the comparison between our two methods and familiar Anti-Virus softwares. The method that provides the best results is Structural feature extraction with results of 0.9787 TPR. This method surpasses all the best Anti-Virus softwares and the knowledge-based method. In addition, we can see that both of structural and knowledge based detection methods capabilities surpass all the Anti-Virus softwares in the task of detecting malicious RTF files.

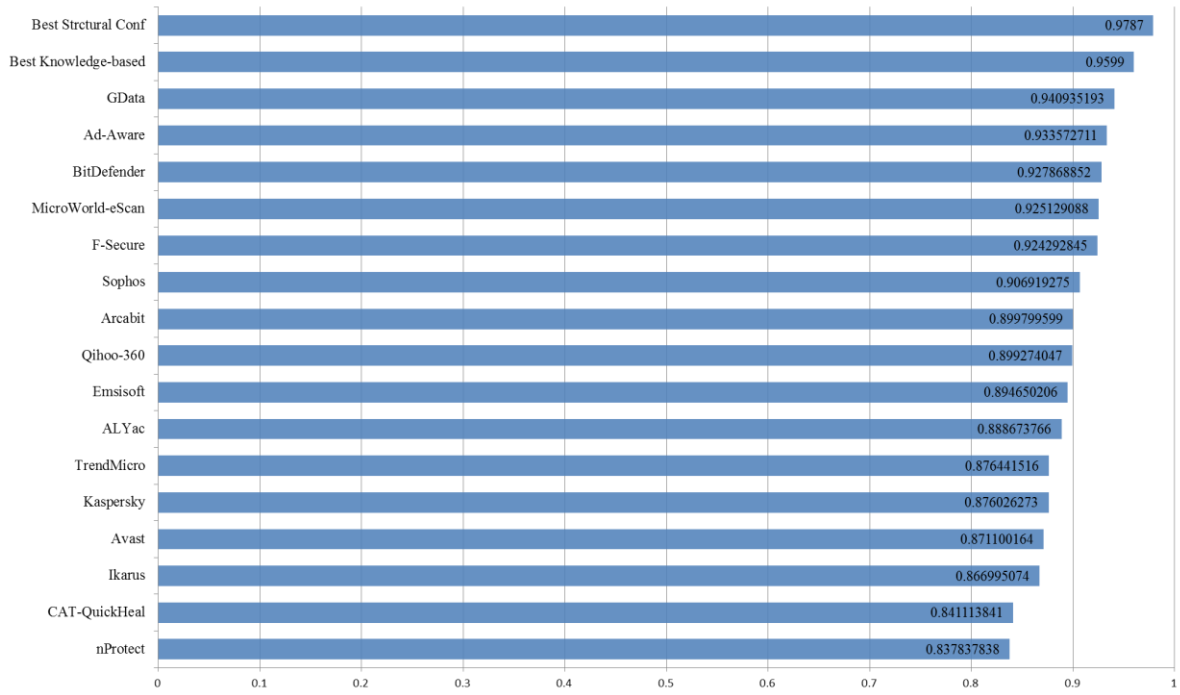


Figure 23. Comparison between accuracy of Structural, Knowledge-Based and Anti-Virus Softwares

VII. DISCUSSION AND CONCLUSIONS

To sum up, we conducted two main theories for detection malicious RTF files. The first one is using a structural feature extraction method and elaborate it. This method was examined in the past (A. Cohen, N. Nissim, L. Rokach, and Y. Elovici, 2016) by creating a tree of hierarchy strings from Top to Bottom. We conducted creation of strings both Top to Bottom, Bottom to Top and single joint. In the experiments, we found out that the Single Joint is a great way for detection malicious files and the Bottom to Top is not. The second theory was learning former attacks, known vulnerabilities and the structure of the file (How it supposed to be built). This method provided a very high detection ratio (approximately 99% correct detection). This method is a new way of detection that surpass even familiar Anti-Virus softwares. It's not depending on specific dataset but a former knowledge that we summed up to features. Thus, we aren't getting overfitting. We believe this method could detect Zero-day attacks and of course known attacks with a high accuracy ratio.

A. Limitations

Our project revealed a new method for detection both known and zero days attacks that surpass even known Anti-Virus softwares. Although, we have to consider some limitations our work had. First of all, during the comparison between the Anti-Virus softwares and our methods, we didn't compare our methods with 0 FPR like the Anti-Virus softwares. Secondly, the Structural feature extraction method used files from the dataset for creation the best features. This situation can lead to overfitting (We tried to handle it by using 10-cross fold validation).

B. Future Work

In our project we examined and displayed a new ways of creation of features for feature extraction (Top to Bottom, Bottom to Top and Singal Joint). More over, we checked the use of former attacks and understanding the file structure for creation of strong and indicative features that supposed to detect known attacks and even zero-day attacks.

Future work supposed to check this methodology on different kind of file like PNG, GIF etc.

VIII. ACKNOWLEDGMENTS

We would like to thank the Malware Lab at the Cyber Security Research Center (CSRC) at Ben-Gurion Univiversity of the Negev, Israel, for supporting this research. Many thanks also to *VirusTotal* for granting us complimentary access to their private services.

IX. MEETINGS TABLE

Date	Participated	Subject	Summary
19.06.16	Aviad Alex Liad	Introduce the project	• Display the project
17.08.16	Aviad Alex Liad	First work meeting	• Start collecting articles • Start readong about RTF and JPG
02.11.16	Nir Nissim Aviad Alex Liad	Second work meeting	• Understand the RTF and JPEG structures • Finding attacks, Vulnerabilities and existing detection Tools
05.12.16	Aviad Alex Liad	Third work meeting	• Presenting the RTF and JPEG structures • Presenting attacks, Vulnerabilities and existing detection Tools • Literature Review
09.01.17	Aviad Alex Liad	Fourth work meeting	• Fine-tuning on Reports • Knowledge-based features and Structural features
24.01.17	Aviad Liad	Fifth work meeting	• Define the programming methods • Fine-tuning on Reports
24.04.17	Aviad Alex Liad	Sixth work meeting	• Finishing the programming phase
24.04.17	Aviad Alex Liad	Seventh work meeting	• Working with Weka • Reports fine-tuning
08.05.17	Aviad Alex Liad	Eighth work meeting	• Results of experiments • Reports fine-tuning

15.05.17	Aviad Alex Liad	Ninth work meeting	• Poster • Results of experiments • Reports fine-tuning
05.06.17	Aviad Alex Liad	Tenth work meeting	• Poster • Fine-tuning on Reports

Table 5. Meetings Table

X. REFERENCES

[ACKS00] ABOU-ASSALEH, TONY ; CERCONE, NICK ; KEˇ, VLADO ; SWEIDAN, RAY: N-gram-based Detection of New Malicious Code Bd. 1, Nr. 1, S. 2–3

[ANWN07] ABU-NIMEH, SAEED ; NAPPA, DARIO ; WANG, XINLEI ; NAIR, SUKU: A Comparison of Machine Learning Techniques for Phishing Detection (2007)

[Chri00] CHRISTODORESCU, MIHAI: Static Analysis of Executables to Detect Malicious Patterns ∗

[CNRE16] COHEN, AVIAD ; NISSIM, NIR ; ROKACH, LIOR ; ELOVICI, YUVAL: SFEM : Structural feature extraction methodology for the detection of malicious office documents using machine learning methods. In: *Expert Systems With Applications* Bd. 63, Elsevier Ltd (2016), S. 324–343

[FCKV00] FORD, SEAN ; COVA, MARCO ; KRUEGEL, CHRISTOPHER ; VIGNA, GIOVANNI: Analyzing and Detecting Malicious Flash Advertisements

[HaVi00] HALLARAKER, OYSTEIN ; VIGNA, GIOVANNI: Detecting Malicious JavaScript Code in Mozilla

[HuLi05] HUANG, JIN ; LING, CHARLES X.: Using AUC and accuracy in evaluating learning algorithms. In: *IEEE Transactions on Knowledge and Data Engineering* Bd. 17 (2005), Nr. 3, S. 299–310 — ISBN 1041-4347

[JoKK00] JOVANOVIC, NENAD ; KRUEGEL, CHRISTOPHER ; KIRDA, ENGIN: Pixy : A Static Analysis Tool for Detecting Web Application Vulnerabilities (Technical Report)

[MaGi00] MAIORCA, DAVIDE ; GIACINTO, GIORGIO: Looking at the Bag is not Enough to Find the Bomb : An Evasion of Structural Methods for Malicious PDF Files Detection — ISBN 9781450317672

[NCGE14] NISSIM, NIR ; COHEN, AVIAD ; GLEZER, CHANAN ; ELOVICI, YUVAL: ScienceDirect Detection of malicious PDF files and directions for enhancements : A state-of-the art survey. In: *Computers & Security* Bd. 48, Elsevier Ltd (2014), S. 246–266

[Otsu15] OTSUBO, YUHEI: O-checker : Detection of Malicious Documents through Deviation from File Format Specifications (2015)

[PeMi93] PENNEBAKER, WILLIAM B ; MITCHELL, JOAN L: *JPEG still image data compression standard*. Bd. 34, 1993 — ISBN 0442012721

[PrST16] PRIYANKA, RAMA ; SAHOO, P K ; TECH, STUDENT M: Scanning Tool For Identification of Image With Malware Bd. 4 (2016), Nr. 1, S. 170–175

[Serv08] SERVER, WINDOWS: Rich Text Format (RTF) Specification Rich Text Format (RTF) Specification (2008)

[SPDB00] SANTOS, IGOR ; PENYA, YOSEBA K ; DEVESA, JAIME ; BRINGAS, PABLO G: N-GRAMS-BASED FILE SIGNATURES FOR MALWARE DETECTION

[ThSc13] THESIS, BACHELOR ; SCIENCE, COMPUTER: Malicious PDF Document Analysis (2013), S. 1–20

[WeKa89] WEISS, SHOLOM M ; KAPOULEAS, IOANNIS: An Empirical Comparison of Pattern Recognition , Neural Nets , and Machine Learning Classification Methods (1989), S. 781–787