# 恶意文档检测研究综述

喻 民[1,2], 姜建国[1,2], 李 罡[3], 刘 超[1], 黄伟庆[1,2], 宋 楠[1,2]

[1]中国科学院信息工程研究所 北京 中国 100093
[2]中国科学院大学 网络空间安全学院 北京 中国 100049
[3]迪肯大学 信息技术学院 吉朗 澳大利亚 VIC 3220

**摘要** 近年来, 以窃取敏感数据、破坏国家重要基础设施为主要目标的高级持续威胁(Advanced Persistent Threat, APT)已经给国家安全带来了严重的威胁。与可执行文件相比, 恶意文档具有涉及领域广、影响范围大、用户防范意识不足、攻击手段灵活多样、难以检测等诸多特点, 已经成为实施 APT 攻击的重要载体。因此有必要关注恶意文档检测已有的研究成果与发展趋势。本文首先对文档类型及其结构进行了解析, 然后阐述了文档的安全隐患、攻击技术以及传播途径等。将当前恶意文档检测方法归纳为静态检测法、动态检测法、动静态结合检测法以及其他相关研究等四类, 分别对各类检测方法的研究状况、进展进行了分析和总结。最后, 提出了当前恶意文档检测研究的性能评价方法, 综述了代表性的数据、检测工具和平台, 并展望了未来的研究方向。

**关键词** 恶意文档; 恶意代码; 检测方法; 性能评价; 特征分析
**中图法分类号** TP309.5 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2021.05.04

# A Survey of Research on Malicious Document Detection

YU Min[1,2], JIANG Jianguo[1,2], LI Gang[3], LIU Chao[1], HUANG Weiqing[1,2], SONG Nan[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
[3] School of Information Technology, Deakin University, Geelong, VIC 3220 Australia

**Abstract** In recent years, Advanced Persistent Threat (APT), which has the primary purpose of stealing sensitive data and undermining critical national infrastructure, has already brought serious threats to national security. Compared with executive files, malicious documents have several unique characteristics, such as wide range of coverage, large scope of influence, insufficient user awareness, flexible and diverse attack methods, and it is a challenge to detect. This has made it an important carrier for implementing APT attacks. Therefore, it is necessary to pay attention to the existing research results and development trends of malicious documents. This paper first analyzes the document type and its structure, and proposes the security risks, attack techniques and propagation paths of the document. The current malicious document detection methods are categorized into four groups: static detection methods, dynamic detection methods, hybrid detection methods and others. The research status and research progress of each field are analyzed and summarized. Finally, the performance evaluation methods, data sets, representative detection tools and platforms of current malicious document detection research are reviewed and proposed, and the future research directions are envisaged.

**Key words** malicious document; malicious code; detection method; performance evaluation; feature analysis

## 1 引言

恶意文档是指能够被用来实现恶意目的的伪装性攻击文档[1]。1993 年, VBA (Visual Basic for Applications, VBA) 语言应用到微软 Office(后称 Office)文档开发中, 不久就出现了宏病毒, 开启了文档攻击的序幕, 1999 年爆发的宏病毒—梅丽莎(Melissa)[2]开创了一种新的攻击载体。随着攻防技术的发展和攻击意图的变化, APT 攻击已经成为当前网络安全的重要威胁, 通过定向攻击的方式, 来实现长期窃取大量敏感数据、破坏国家重要基础设施等行为, 严重危害国家安全。

2013 年, 瀚海源安全公司(现已被阿里公司收购)就发现并报告了一例利用 WPS (Word Processing

**通讯作者:** 喻 民, 博士, 高级工程师, Email: yumin@iie.ac.cn。
本课题得到中国科学院青年创新促进会(No.2021155)资助。
收稿日期: 2019-06-03; 修改日期: 2019-09-06; 定稿日期: 2021-03-05

System, WPS) 软件 0day 漏洞, 通过钓鱼邮件对中国政府进行定向攻击的事件。2014 年起, 针对叙利亚地区的 APT 攻击采用 Word 文档配合 JavaScript 脚本的方式实施“水坑攻击”。2015 年, 攻击者利用包含恶意宏病毒的空白 Word 文档绕过垃圾邮件过滤系统, 对目标人群进行定向攻击; 同年, 针对乌克兰电网的 APT 攻击利用恶意 Word 文档实施“鱼叉攻击”, 定向破坏目标的电力网络系统。2016 年, 发现勒索软件 Locky 的新变种利用 Word 文档作为传播载体, 从指定的 URL (Uniform Resource Locator, URL)下载勒索软件主体[3]; 同年, 针对巴基斯坦与中国的科研院所、军事院校和外交官窃取文件与军事情报的 APT 攻击“丰收行动”, 也是以邮件附件形式发送恶意 Word 文档作为后续持续攻击的关键步骤。2017 年, 攻击者利用包含恶意宏和 Powershell 的文档对韩国平昌奥运会网络系统进行攻击, 导致奥运会网站瘫痪。2018 年, 网络犯罪团伙利用武器化 Word 文档传播新的恶意软件, 攻击各种金融机构, 并逃避 Windows 防御, 研究者还发现攻击者利用恶意 YouTube 视频缩略图嵌入 Word 文档的方式执行恶意 JavaScript 代码[4]。2019 年以来, 恶意文档攻击技术逐渐成熟, 以 PDF(Portable Document Format, PDF)、Office 为代表的恶意文档已经成为实施 APT 攻击的重要载体。

PDF 和 Office 是当前世界上使用最为普遍的文档类型。据统计, Office 市场渗透率高达 97.31%, 也是我国党政机关工作人员最常用的软件, 一旦感染成功, 扩散十分迅速。2017 年底, 360 威胁情报中心发布的《2017 年鱼叉攻击邮件研究报告》指出, 攻击者在邮件中携带最主要的文档为 Office, 占比高达 65.4%, 其次为 RTF, 占比达到了 27.3%。卡巴斯基 2018 年第三季度安全报告显示[5], Office 在软件漏洞利用统计中以 70%高居第一, 遥遥领先其他软件。仅 2019 年, CVE (Common Vulnerabilities and Exposures, CVE) 发布 PDF、Office 的漏洞达 100 多个, 其中不乏 CVE-2019-1786, CVE-2019-0541, CVE-2019-0797, CVE-2019-0953 等高危漏洞, 受此影响的还有兼容此类软件的浏览器。

恶意文档已经成为实施 APT 攻击的重要载体, 由于反病毒技术快速发展以及免费安全软件在全球的高度普及, 可执行文件(Portable Executable, PE)的传播变得十分困难, 越来越多的黑客更加倾向在 APT 攻击中利用 PDF、Office 等文档作为实施攻击的第一步。与 EXE、DLL (Dynamic Link Library, DLL)等 PE 文件相比, 人们普遍将文档文件看作高效、便捷与安全的信息交互载体, 攻击者正是利用这种普遍认知, 大肆利用恶意文档进行攻击, 实现窃取用户信息、恶意破坏系统、远程控制用户计算机等目的。同时, 为提供丰富的功能支持, 这些文档不仅具有复杂的文档结构, 并且支持嵌入脚本语言(比如, JavaScript、 VBA 等), 在文档中还可以内嵌其他类型的文件。复杂的文档结构为恶意代码提供了隐藏和存储空间, 支持脚本语言则为恶意代码的执行提供了可能, 内嵌的文件内可能存在恶意代码或本身就是恶意程序, 而这些原本为用户提供丰富功能的特性为恶意文档攻击提供了条件。

近年来, 恶意文档的检测已经成为热门研究领域, 许多新的研究思路和方法不断被提出。本文正是以此为背景, 对恶意文档检测的现有成果进行评述, 对相关的研究思路、方法和开源工具进行比较和分析, 并指出未来的研究方向。本文的主要贡献: 1) 全面、系统、深入地论述了恶意文档检测技术的实现原理和研究进展; 2) 分类阐述了文档存在的安全性问题; 3) 归纳了当前恶意文档检测的评价体系、数据集以及开源产品; 4) 对恶意文档检测技术的发展进行展望。

本文其他章节的组织结构如下: 第 2 节介绍文档的分类、结构等背景知识; 第 3 节分析文档存在的安全隐患、攻击技术以及传播途径; 第 4 节对恶意文档的检测技术进行全面、完整的综述, 特别的是, 本文作者提出了一种基于多层抽象的恶意文档统一检测方法; 第 5 节归纳了当前恶意文档检测的评价指标、数据集以及开源检测工具、辅助分析工具与在线检测平台; 第 6 节对全文进行总结, 并就恶意文档检测技术今后的研究方向进行了展望。

## 2 背景知识

恶意文档的检测, 首先对文档结构进行深入剖析, 包括物理结构、逻辑结构等。由于 PDF 和 Office 文档在全球市场占有绝对的主流地位, 具有充分的代表性, 因此, 本节将详细介绍 PDF 和 Office 文档的物理和逻辑结构, 最后简要介绍其他文档格式。

### 2.1 PDF 文档

PDF 文档的结构可以从两个角度进行分析, 一方面是 PDF 文档实际存储在硬盘空间中的数据分布结构, 称为物理结构; 另一方面从 PDF 文档阅读器解析文档的逻辑顺序角度分析文档的结构, 称为逻辑结构。分析物理结构, 我们可以获知文档信息在各个组成模块的分布情况, 方便解析文档时从相应的位置提取所需的信息。通过文档的逻辑结构, 可以获

知文档各个组成模块的逻辑依赖顺序和关系, 从而设计文档的解析和信息提取顺序。本小结将从物理结构和逻辑结构分别介绍。

### 2.1.1 物理结构

PDF 文档是由一组相互连接的对象分层次进行构建的, 根据 Adobe PDF Reference[6], 其物理结构由四个基本部分组成, 如图 1 所示。

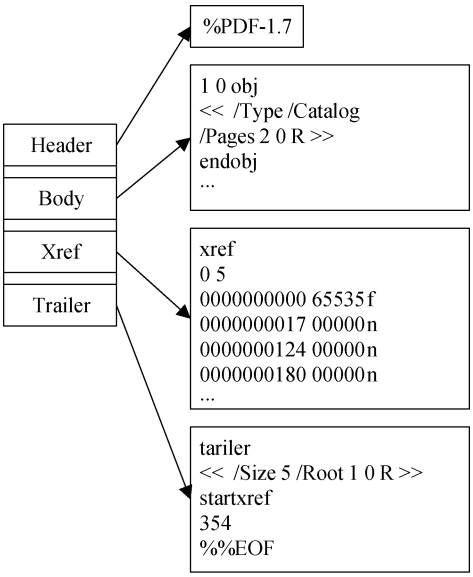


**图 1 PDF 文档物理结构**

**Figure 1 Physical structure of a PDF document**

(1)文件头(Header): PDF 文档的类型标识位于文件头, 以“%PDF-”为起始, 后接该文档所遵循的版本, 如: “%PDF-1.7”, 表示该文档符合的是 PDF 1.7 版本规范。

(2)文件体(Body): 文档存储的主要内容都在文件体中, 由多个间接对象(Indirect Object)构成, 间接对象可存储文字、图片等内容, 包括文字的字体、颜色、大小, 图片的内容、布局、位置等, 也可记录对象之间的依赖关系和展现顺序等信息。间接对象以对象号(Object Number)、生成号(Generation Number)和关键字“obj”表示起始, 以关键字“endobj”表示结束。中间内容以“<<”、“>>”为开头结尾的字典(Dictionary), 以关键字“stream”、“endstream”为开头结尾的字符流组成。字典中的内容以 key-value 形式存储, 可以看作是属性-属性值的列表, 字符流一般存储文档中的文字内容, 图片信息, 内嵌 JavaScript 代码等内容。PDF 文档在存储这些内容时, 一般会采用现有的压缩方法进行压缩, 以此节省存储空间和文档传输成本。

(3)交叉引用表(Xref): 以关键字“xref”为起始, 此部分主要记录了各个对象在文档中相对于起始位置的偏移地址, 并以字节为单位。

(4) 文件尾 (Trailer): 以“trailer”为起始, 以“%%EOF”为结尾, 主要存储两个重要的关键信息, 一个是以属性关键字“Root”标识的根对象; 另一个是以关键字“startxref”标识的交叉引用表相对于文档起始点的字节偏移位置。

### 2.1.2 逻辑结构

PDF 文档大体上是一种树形的逻辑结构, 其根节点 Catalog 是 PDF 文档物理结构和逻辑结构的连接点, 如图 2 所示。

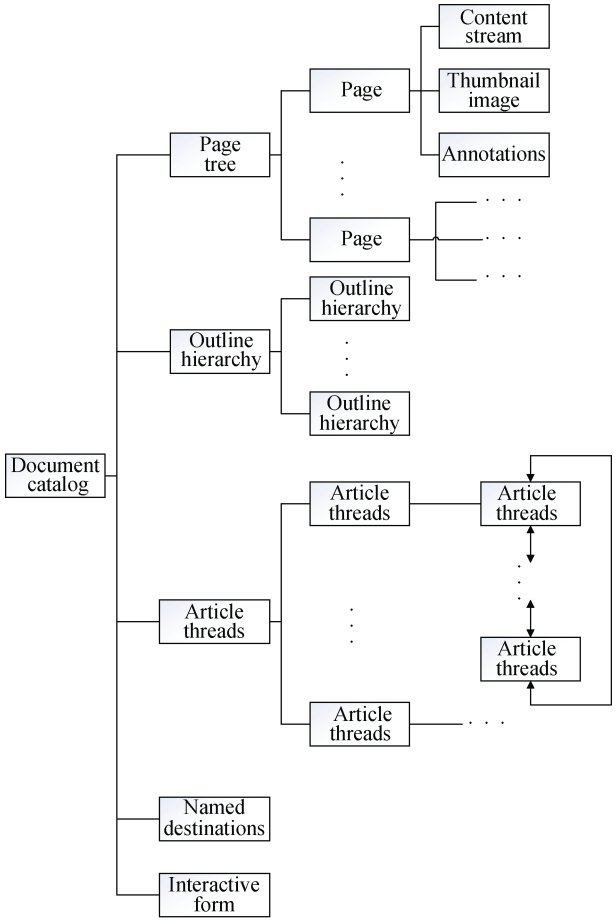


**图 2 PDF 文档逻辑结构**

**Figure 2 Logical structure of a PDF document**

文档阅读器打开一个 PDF 文档时, 首先会从 Trailer 的 Root 字段定位 Catalog, 通过该节点去解析页、目录、链接信息等, Catalog 包含的信息非常多, 包括 Page tree、Outline hierarchy、Article threads 等, 其中, Page tree 是对 PDF 文档中所有页面的描述集合, 用于组织所有的 Page 对象; 而 Page 对象则用于具体描述一个 PDF 页面的属性、资源等信息, 其他条目项目信息可以参考 Adobe PDF Reference。

## 2.2 Office 文档

与 PDF 文档类似, Office 文档也需要从实际的物理存储结构了解 Office 文档各组成部分的分布, 到相应模块处提取信息; 从文档的逻辑顺序获知各组成部分的逻辑依赖顺序与关系, 从而明确文档解析的顺序。目前 Office 文档可分为 Office 2003 和 Office 2007 版本, 每个版本包括 Word、Excel、PowerPoint 等应用, 为方便介绍, 本小节以 Word 作为 Office 文档的代表, 分别介绍 Word 2003(DOC)和 Word 2007(DOCX)的物理结构和逻辑结构。

### 2.2.1 物理结构

(1)DOC 文档

DOC 文档实质上是一个基于二进制的复合文档, 采用了对象连接与嵌入(Object Linking and Embedding, OLE)技术存储数据, 包括文本、图像、音频、视频、表格等, 如图 3 所示。

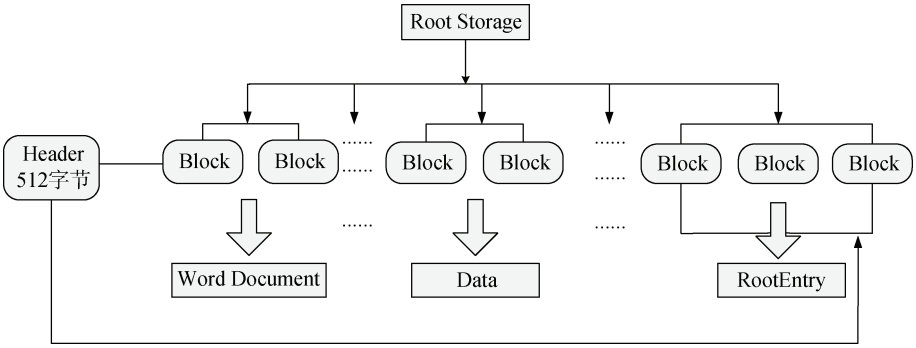


**图 3 DOC 文档的物理结构**

**Figure 3 Physical structure of a DOC document**

DOC 文档由文件头(Header)和扇区(Sector)组成。文件头主要包括文档版本说明、扇区大小等相关信息。扇区的种类有存储器、目录流、扇区分配表、短扇区分配表、剩余扇区分配表五种, 具体信息参考文献[7], 值得注意的是, 文档中较大的数据流, 很有可能被分配到不同的扇区中, 并且存储的扇区可能是无序的, 扇区分配表最主要的作用就是记录了扇区的实际顺序, 这也是 DOC 文档特征提取中需要注意的地方。不难发现, DOC 文档物理结构复杂, 存在一定的安全隐患。

(2)DOCX 文档

DOCX 文档遵循的 OOXML(Office Open XML, OOXML)标准, 在 2008 年通过 ISO 标准认证时更改为 OXML(Open XML, OXML)标准[8], 新标准使用 ZIP 压缩格式, 核心是使用 XML 结构和 ZIP 容器。将 Office 文档中类似于 PDF 文档的对象属性存放于 XML 文件中, 每个 XML 可以对比一类 PDF 文档对象, 非常容易与 PDF 文档抽象成同一种表现形式。如图 4 所示, OXML 文件由多个部件组成, 部件可以包含任何类型的数据, 包括用户定义 XML、嵌入代码/宏、文档属性、图表、图片、音视频文件、注释等。不同部件之间利用关系部件通信, 这使得每个部件既独立但整体又高度融合, 所以 OXML 文件格式具有很好的健壮性、安全性和高效性。

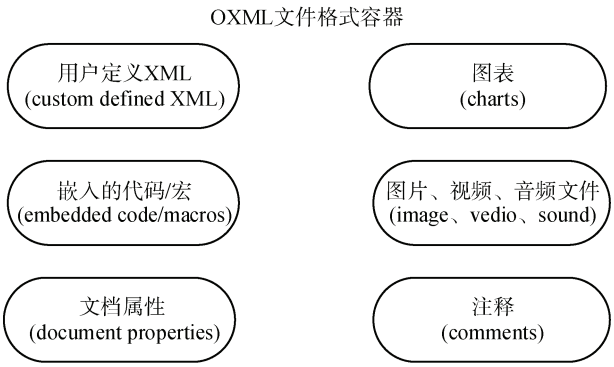


**图 4 OXML 文件结构**

**Figure 4 Structure of OXML**

### 2.2.2 逻辑结构

(1)DOC 文档

一个典型的 DOC 文档逻辑结构如图 5 所示。其中, Data 存储图片数据信息, 1Table 存储数据表, CompObj 存储 Word 版本等通用信息, WordDocumnet 存储文字和格式化信息, SummaryInformation/DocumentSummaryInformation 存储摘要信息, ObjectPool 是一个对象池, 存储 DOC 文档中嵌入的图像、声音或者其他对象, Macros 存储宏的相关内容。目前, 恶意代码已经不仅仅嵌入在 Macros 里面, 在一些新型攻击中, 攻击者已经将恶意代码嵌入在 WordDocument、SummaryInformation、DocumentSummaryInformation 等容易被现有文档检测方法忽

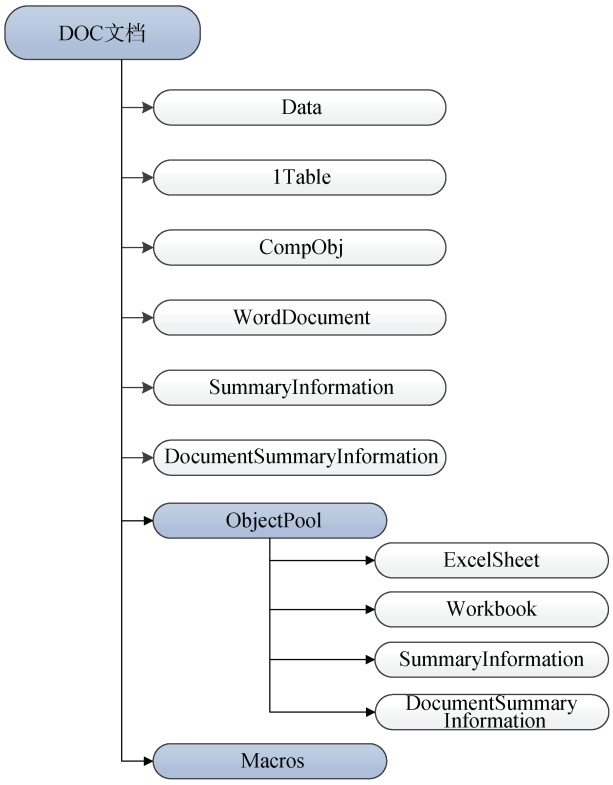


**图 5　DOC 文档的逻辑结构**

**Figure 5　Logical structure of a DOC document**

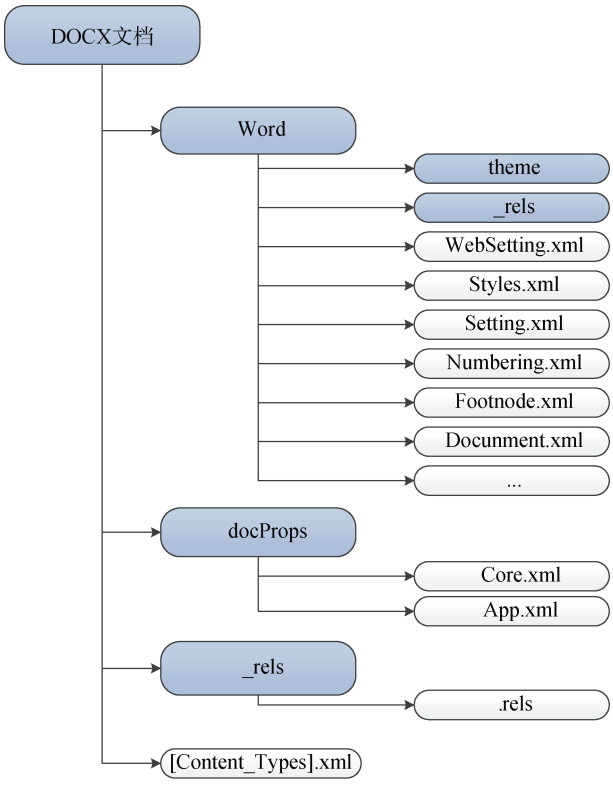


**图 6　DOCX 文档的逻辑结构**

**Figure 6　Logical structure of a DOCX document**

略的文件中, 这也是造成当前检测方法准确率不高的一个重要因素, 为了提高代码提取的准确性, 这些区域也将是今后研究的重点。

(2)DOCX 文档

DOCX 文档逻辑结构比较透明, 解压缩后可以看到所有的文件, 如图 6 所示。其中, Word 存储了文档的主要内容, 比如, 文档的文本内容、文本样式、字体设置存储在 Document.xml 文件, 脚注信息存储在 Footnode.xml 等; _docProps 包含多个文件, 比如 app.xml 文件记录了文档的属性, 类似 DOC 文档的 DocumentSummary Information, core.xml 文件记录了文档创建时间和时间, 类似 DOC 文档的 SummaryInformation 文件; rels 文件夹是文档解析的入口; Content_types 则记录了该文件以外所有部件包含的文件类型。不难发现, DOCX 文档逻辑结构呈现出层次性, 再结合每个 XML 文件中节点的属性特征, 不同文档会表现出差异, 这也为文档结构特征提取提出了挑战。

## 2.3　其他文档格式

富文本格式(Rich Text Format, RTF)是一个带有发布规范的专有文档文件格式, 微软公司使用其与微软产品进行跨平台文件交换, 大多数文档阅读器支持 RTF 格式导入、导出或直接编辑。因此, RTF 通常作为文字处理软件和操作系统之间的“通用”格式, 在全球使用范围较广, 且大多数读取 RTF 文件的应用程序默认忽略未知的 RTF 控制字, 增强了 RTF 文档的通用性, RTF 文件的语法可参见微软公司发布的 Rich Text Format (RTF) Specification v1.9[9]。RTF 文件的物理结构包括文件头 <header> 和文档区 <document> 两大部分。RTF 文档区呈现层次结构: 文档>节>段落>对象(正文、图片、字体、OLE 对象等), 文档、节、段等层次主要定义各种格式属性, 对象层除了描述具体对象的属性外, 还包括对象的实质内容。RTF 的安全隐患主要体现在以下两点: 一是 RTF 虽然不支持宏, 但是 Word 允许使用 RTF 扩展名的标准 DOC 文件运行其中包含的宏; 二是 RTF 可以嵌入能存储可执行文件和脚本的恶意 OLE 对象, 从而利用 Word 或 Flash 等存在的漏洞达到攻击目的。

WPS 系列办公软件[10]1988 年诞生于中国, 目前最新版 WPS Office 2019 符合国际标准 OXML 和国内标准 UOF 2.0。这是一种以 XML 为基础并以 ZIP 格式压缩的电子文件规范, 支持文档、表格、备忘录、幻灯片等文件格式。支持微软 VBA 编程环境, 可以直接打开带有宏代码的 Office 文档。不过 WPS 软件的更新周期较短, 存在对象链接(Object Linking, OL)攻击和动态数据交换(Dynamic Data Exchange, DDE)

攻击, 需要用户注意 WPS 应用程序的警告提示。

其他文档格式还包括 MHTML(MIME Encapsulation of Aggregate HTML Documents, MHTML)[11], 可作为载体发起“鱼叉攻击”和“水坑攻击”, 文档内可包含恶意的 VBA 代码, 是 APT 攻击中常见的恶意样本类型, 且该类型恶意样本有逐渐增多的趋势。ARJ(Archived by Robert Jung) [12]、IQY(Internet Query File)[13]、PUB(Microsoft Publisher)[14]、SettingContent-ms[15]。其中, ARJ 文档通过嵌套可执行文件或可执行的屏幕保护程序来感染计算机; IQY 和 PUB 文档通常嵌入到 PDF 文档中进行攻击, IQY 文件默认在 Excel 中打开, 通过网络下载的数据作为感染媒介, PUB 文档是一个特殊的 Microsoft 办公应用程序。SettingContent-ms 格式是在 Windows 10 中引入的, 用于为各种 Windows 10 设置页面创建“快捷方式”, 实际上, 它们就是一些 XML 文件, 用于存放 Windows 10 设置二进制文件的路径。攻击者将 SettingContent-ms 文件中 DeepLink 标记指向其他恶意程序或者可以执行的脚本代码(比如 Powershell), 并诱导用户在 Windows 10 下执行该文件, 或者通过其他载体(Office、PDF)执行该文件, 都可能执行恶意代码从而导致电脑被控制。

## 3 恶意文档攻击

攻击者在构造恶意文档前, 需要对文档的安全隐患、攻击技术、传播途径等进行研究, 达到准确、高效攻击的目的。

### 3.1 安全隐患

文档的安全隐患是指文档可被利用实现攻击的安全脆弱性, 包括存在的漏洞、安全机制、用户安全意识等方面。

#### 3.1.1 畸形文档

畸形文档是指文档的结构、格式等属性不符合文档标准的要求, 但是仍然能够被文档阅读器解析。造成这个现象的主要原因是各类文档阅读器在设计之初, 并没有考虑太多安全方面的问题(如特殊字符或非法格式的处理), 以兼容性、便携性作为产品开发的主要目的, 比如由于 Word 文档的兼容性, 攻击者可以隐藏多种恶意文件(视频、图片、音频、文档等)和恶意代码(JavaScript、Macro、Powershell 等), 由于这些功能符合 Word 文档标准要求, 杀毒软件会忽视这些恶意行为。相比于 Word, Adobe Reader 对格式错误的容忍度更高, 畸形 PDF 文档更为常见, 主要体现在以下三个方面:

(1)按照 Adobe Reader 官方文档的规定: 文档对象必须以数字标号、生成号、关键字“obj”开头, 以关键字“endobj”结尾, 这也是文档阅读器解析 PDF 文档的重要依据, 但是文档阅读器在实现对象的划分时却可以不遵循该依据。

(2)PDF 文档中的交叉索引表也有很高的容错性, 交叉索引表中的信息可以允许错误, 甚至丢失交叉索引表也不会影响 PDF 文档的解析。Adobe Reader 阅读器可以正常显示文档, 这是因为 Adobe Reader 在解析文档时会进行纠错处理, 并自行生成新的交叉索引表。

(3)具有相同对象序号的对象会发生覆盖, Adobe Reader 在解析 PDF 文档时, 当解析到相同对象序号的对象时, 先出现对象的内容会被后出现对象的内容覆盖。

基于以上三点, 导致了如图 7 所示的畸形文档结构也能被 Adobe Reader 正常解析, 但是在文档打开时会自动执行 JavaScript 代码。可以看到红色部分的对象以“xxxxxx”代替关键字“endobj”作为一个对象的结束标识, 这导致了该红色区域的对象无法被现有的开源 PDF 文档解析工具识别。同时, 该红色区域内也出现了对象序号为 1 的 Root 对象, Adobe Reader 阅读器在解析该 PDF 文档时会识别该红色部分的 Root 对象作为新的根对象, 覆盖之前的根对象信息, 而新的根对象信息中包含了 JavaScript 代码的执行信息。

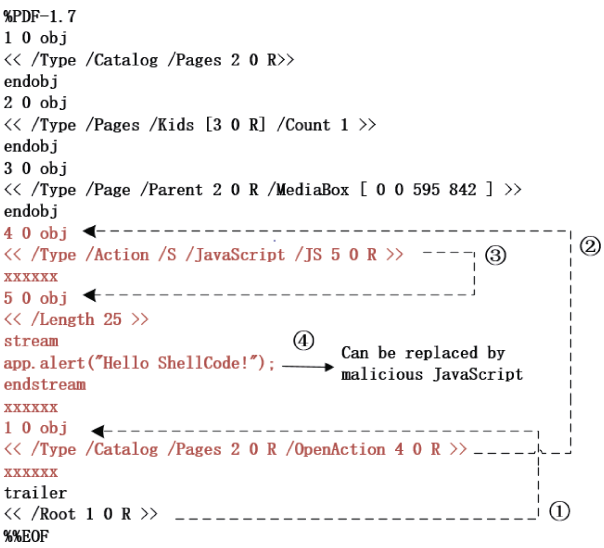


**图 7 畸形 PDF 文档示意图**

**Figure 7 The instance of a malformed PDF document**

本示意图中没有出现交叉索引表, 这导致部分依赖交叉索引表定位各个对象的开源工具, 或是直接无法正常运行, 或是无法从交叉索引表中获知红色区域对象存在。但是, 当文档中存在交叉索引表时,

需要的前提条件是红色区域的对象必须出现在交叉索引表之前, 本实例的 JavaScript 代码才会执行, 因为 Adobe Reader 将出现在交叉索引表之后的对象判别为无效对象。由于以上原因导致了该示例中的 JavaScript 代码执行, 而且当前的开源解析方法无法提取到 JavaScript 代码等有效特征, 因为这类有效特征都隐藏在了图中红色区域的对象中, 而这类对象无法被现有开源解析方法所识别。导致利用这些开源代码对恶意文档进行检测的方法无法提取有效的恶意特征, 最终导致检测失效。

### 3.1.2 文档漏洞

PDF、Office 等各类文档开发时间久, 功能复杂, 经过了多次的版本更替与兼容, 存在大量的漏洞。漏洞一般可分为 0day 漏洞和非 0day 漏洞, 0day 漏洞具有强大的杀伤性和破坏力, 但对攻击者技术要求较高, 如 CVE-2017-8759, 该漏洞就是 2017 年发现的一个 0day 漏洞, 它主要利用了简单对象访问协议(Simple Object Access Protocol, SOAP) 和网络服务描述语言(Web Services Description Language, WSDL)分析器代码注入漏洞, 该漏洞可以在解析 SOAP 和 WSDL 定义的内容时注入任意代码, 并且被攻击者利用了至少一个月后才被正式公开; 非 0 day 漏洞同样具有很强的破坏性, 当漏洞被公开后, 攻击者就会对漏洞细节更加清晰, 虽然补丁也会同时发布, 但是用户往往选择忽视。攻击者利用这些漏洞进行 APT 攻击, 发动僵尸网络、勒索病毒等传播, 窃密重要机密信息等。攻击者正在不遗余力地挖掘这些漏洞, 最近几年曝光的各类文档漏洞越来越密集, 文档漏洞攻击仍然会不断持续, 也会不断出现新的攻击手段, 主要分类如下表所示。

### 3.1.3 安全机制的脆弱性

PDF、Office 等文档的广泛应用为用户带来了极大的便利, 但是这些文档的安全机制并不复杂, 比如 Word 的安全机制主要分为三个方面: 宏安全策略保护、宏安全等级保护和信任域保护, 不幸的是, 这三种安全机制都被证明可以被绕过[16], 更糟糕的是, 攻击者往往会采用社会工程学的手段诱使攻击者打开宏文件或降低宏权限, 并不需要任何复杂的技术手段, 就可以绕过安全防护机制。

### 3.1.4 用户安全防范意识薄弱

据统计, 2017 年以来公布的涉及 PDF、Office 等文档的高危漏洞达 100 多个, 并且根据安全公司统计发现, 漏洞补丁发布半年内, 用户修复比例不足 70%, 很多用户虽然收到补丁通知, 但是由于安全意识缺乏等原因, 往往忽视更新, 这给文档攻击者提供了极大的便利。目前, 网络攻击的趋势越来越趋近于多种攻击技术协同攻击, 而许多攻击者将恶意文档攻击作为开启攻击的第一步[17], 攻击者往往向特定目标发送包含恶意文档的钓鱼邮件, 并诱使用户下载附件并打开。从大量案例中看出, 即使是警惕性比较高的人员, 也很容易中招。

**表 1 文档漏洞分类表**

**Table 1 Document vulnerability classification table**

| 漏洞类型 | 典型漏洞 CVE | 特点 |
|---|---|---|
| 远程代码执行漏洞 | CVE-2019-0953<br>CVE-2019-0947<br>CVE-2018-8574 | 当 Word 软件无法正确处理内存中的对象, 存在远程代码执行漏洞。 |
| 逻辑漏洞 | CVE-2019-7027<br>CVE-2017-0199<br>CVE-2017-8570 | 无需用户交互, 打开 Word 文档就可以通过 hta 脚本执行任意代码, 或者在 PDF 阅读器中执行越界写入操作, 成功率较高。 |
| EPS (Encapsulated PostScript)漏洞 | CVE-2017-0261<br>CVE-2017-0262 | 利用 EPS 文件执行漏洞, 精心构造后的样本可以绕过微软的相关安全机制。 |
| .Net Framework 漏洞 | CVE-2017-8759 | 该漏洞本质上是一个.Net Framework 漏洞, 可以集成到 Word 等 Office 文档中。 |
| 内存破坏漏洞 | CVE-2017-11826<br>CVE-2015-1641<br>CVE-2012-0158 | 这类漏洞可以破坏内存从而执行攻击者精心准备的 Shellcode。 |
| 类型混淆漏洞 | CVE-2017-11826 | OOXML 阅读器在处理 DOCX 文档的 Word/document.xml 时, 没有正确地验证标签对象是否闭合, 造成类型混淆, 可造成任意代码执行嵌入恶意代码类攻击。 |

## 3.2 攻击技术

恶意文档的攻击技术主要包括以下几种: 恶意代码攻击、对象嵌入攻击、文档漏洞攻击以及远程链接攻击。

### 3.2.1 恶意代码攻击

攻击者最常使用的就是嵌入恶意代码进行攻击, 因为 PDF、Office 等文档都支持强大的脚本功能, 这种特性允许攻击者利用阅读器在解析文档时的漏洞, 执行嵌入在文档内的恶意代码, 从而实现攻击。同时攻击者还会使用代码混淆来隐藏恶意代码, 使其无法被基于签名的检测器识别, 并降低人工分析的可读性, 主要包括以下几种:

(1)利用脚本语言直接执行恶意指令, 这种攻击方法在恶意 PDF 文档和恶意 Office 文档中都比较常见。在恶意 Office 文档实现这种攻击时一般会利用宏病毒即 VBA 代码对字符串进行拼接, 形成

Powershell 指令的参数, 最后调用 Powershell 来实现远程恶意程序的下载, 反弹 shell 等恶意行为。

(2)利用脚本语言间接执行 Shellcode, 这种攻击方法常见于恶意 PDF 文档, 利用 PDF 阅读器对 JavaScript 代码在应用程序编程接口(Application Programming Interface, API)实现上的漏洞, 结合堆喷射技术(Heap Spraying)和面向返回的编程(Return-Oriented Programming, ROP)等方法使得 Shellcode 代码能够在内存中执行。常见的流程是恶意样本在劫持了控制流后, 先会运行 ROP 链, 从而修改 Shellcode 所在内存区的执行属性, 当内存属性修改成功后再去执行 Shellcode。

(3)在 Office 文档方面, 还可以利用 DDE 发起攻击, 使用该方法可以绕过 Office 宏限制, 而且不需要利用 Office 漏洞就可以执行命令。由于 DDE 本身是 Office 的合法功能, 所以大多数安全软件不会阻止 DDE 字段运行, DDE 攻击往往需要与用户进行交互, 一旦用户确认提交后, 文档里嵌入的恶意代码便会开始执行, 通常是远程下载恶意木马等病毒。

### 3.2.2 对象嵌入攻击

PDF、Word 等文档都支持嵌入其他类型的文件, 例如 HTML、JavaScript、Office、PDF、OLE 对象等。攻击者可以使用此功能将恶意文档嵌入良性文档中, 利用其他文件类型的漏洞来执行恶意攻击; 或者通过在文档中嵌入恶意的可执行文件, 利用文档中的 JavaScript、VBA 等脚本使得嵌入的可执行文件执行。以 OLE 对象嵌入为例, 利用 OLE 技术, 可以允许应用程序共享数据, 其存在虽然丰富了文档的功能, 但也给文档带来了安全隐患。基于 OLE 对象嵌入的攻击方式可以分为两种, 一种是基于 OLE 对象自身的攻击, 另一种是基于 OLE 解析漏洞的攻击。

(1)一个OLE对象可以包含任何文件或者命令语句, 一旦用户双击这个对象, 里面的文件或者命令就会自动执行。为了增加攻击的成功率, 攻击者通常将嵌入的对象缩小成一个点或者用户信任的图标来掩饰自身, 并结合社会工程学的手段, 诱导用户进行点击。

(2)基于 OLE 解析漏洞的攻击, 大部分的 Office 高危漏洞, 甚至是 0day 漏洞都涉及 OLE, 类别覆盖从内存破坏到逻辑漏洞等, 在初始化 OLE 对象时, 攻击者通常利用 IPersistStorage::Load 和 IOleObject::DoVerb 这两个重要的攻击载体实现对文档的攻击[18]。而且现在已出现利用 OLE 漏洞进行逃避检测的攻击方法, 需要被重点关注。

### 3.2.3 文档漏洞攻击

文档漏洞攻击是指利用文档的漏洞进行网络攻击, 一般通过远程下载恶意程序或直接执行恶意代码等方式实施攻击, 带有信息窃取、敲诈勒索等强烈意图。以 2017 年爆发的 Office 漏洞 CVE-2017-0199 为例, 该漏洞在 2017 年 3 月被公开, 在 4 月份集中爆发, 这也说明漏洞即使被公开也会对网络安全造成严重的威胁。CVE-2017-0199 是 Office 的 OLE 处理机制实现上存在的一个逻辑漏洞, 在处理内嵌 OLE2LINK 对象时, 通过网络更新对象时未正确处理 Content-Type 所形成的一个漏洞。攻击者可利用此漏洞构造恶意 Office 文件, 当用户打开特殊构造的恶意 Office 文件后攻击者可以在用户系统上执行任意命令, 从而控制用户系统。该漏洞影响非常广泛, 覆盖 Office 所有版本, 且构造恶意文档相对其他漏洞更加简单。它的载荷也非常广泛, 从常用的信息盗取木马、远控木马和下载器, 到敲诈勒索都有涉及, 相比于传统的恶意程序更具有迷惑性, 攻击者通过“鱼叉”或“水坑”攻击方式, 并结合社会工程学手段, 精心构造文档名及伪装内容, 攻击成功概率大大提升。

### 3.2.4 远程链接攻击

攻击者利用一些规范支持的指令或网络链接地址可以使恶意文档访问远程链接下载恶意代码。2013 年, Hamon[19]提出使用表单提交的方式从 PDF 文档中执行恶意代码, 利用 Adobe Reader 使用/SubmitForm 命令将 PDF 表单从客户机提交到特定的服务器, 生成一个表单数据格式(Form Data Format, FDF)文件, 将数据发送到指定的 URL, URL 的响应暂时存储在%APPData%目录中, 会在默认 Web 浏览器中自动弹出。攻击可以通过对恶意网站的简单请求来执行, 恶意网站会自动弹出到 Web 浏览器中, 然后利用用户 Web 浏览器中的漏洞。而相应的安全机制如 Adobe Reader X 的保护模式、Internet Explorer 的 URL 安全区域管理器等可以通过更改相应的注册表项轻松禁用, 可以通过更改用户权限实现。

另外还可以使用统一资源标识符(Uniform Resource Identifier, URI)地址进行攻击, URI 可以无限制地指向远程定位的任何类型文件, 包括可执行文件和非可执行文件。一个类似模仿攻击的场景是从一个良性的 PDF 文档启动一个恶意的 PDF 文档。而如果利用 URI 指向一个可执行文件, 则经常与上述表单提交攻击配合使用: 可执行文件用来更改安全机制的配置, 之后利用良性 PDF 发送表单指向恶意服务器, 获取目标的阅读器信息后通过生成对应版本漏洞的恶意 PDF 文档进行攻击。

### 3.3 传播途径

恶意文档的传播方式主要有以下几种[20]:

路过式传播: 攻击者将恶意文档上传到网络上, 利用网站的 Web 安全漏洞使得当有用户访问某些网页时, 恶意文档被自动下载到本地计算机, 是“水坑攻击”的最主要手段。

邮件传播: 攻击者利用电子邮件进行的网络攻击越来越具有针对性, 对组织的破坏性更大, 这类有针对性的攻击一般称为“鱼叉攻击”。通常针对个人或者小群体, 利用个人隐私、社会工程学等制作非常可信的信息, 目的是诱使收件人打开附件或访问不安全的网站, 这些附件大多是精心设计的恶意文档。虽然此类攻击成本高, 但是攻击成功概率高, 潜在危害也更大, 成为恶意文档传播的主要途径。

社工学传播: 通常也会采用邮件附件的方式传播恶意文档, 不同的是被攻击者往往是特定的, 邮件的内容更具针对性。邮件的发件人, 附件的文档内容等都会经过精心的设计或伪装, 恶意文档的制作也往往利用了新的漏洞和攻击方法。

## 4 恶意文档检测

恶意文档的检测最早利用了传统基于签名或启发式的病毒检测方法[21], 通过对特征片段(如机器码序列字符串)计算哈希值, 构建指纹库进行匹配的方式进行检测, 这种检测方法需要经常更新指纹库信息, 对新样本不具备检测能力, 检测的准确性完全依赖指纹库。随着技术的发展, 恶意文档检测的研究主要包括静态检测方法、动态检测方法、动静结合检测方法, 以及其他相关研究。

### 4.1 静态检测方法

静态检测方法通常是指在不打开文档的前提下, 通过分析文档嵌入的代码、文档内容、结构等特征判断文档性质。主要包括基于统计分析的检测方法、基于内容特征的检测方法、基于结构特征的检测方法和基于多层抽象的检测方法。

#### 4.1.1 基于统计分析的检测方法

Schultz 等人[22]在 2001 年提取了恶意负载中的字节序列特征, 并利用已知的良性样本和恶意样本训练贝叶斯分类器, 实验结果表明该方法取得较好的检测性能, 虽然该论文提取的特征较为粗糙, 但为今后基于统计分析的恶意文档检测方法奠定了发展的基础。统计分析法的主要原理是将文档看作一个连续的字节序列, 提取连续字节作为分析特征, 比如提取其 n-gram 作为分析特征[23], 计算 n-gram 的马氏距离[24]等方式作为判定恶意文档的依据。

文献[25]生成了一个良性和一个恶意 5-gram 的检测模型, 同时作者考虑了 DOC 文档不同扇区的字节序列会对检测结果产生影响, 所以给予不同扇区以不同的权重, 最后根据待测文档与良性模型和恶意模型的相似性程度对文档性质进行判断, 虽然该方法更加细粒度, 但是判断的准确率仍然依赖于训练样本的精度, 而且如果嵌入的恶意代码过短, 则很容易被模型忽视掉。文献[26]利用 5-gram 计算了文档内嵌入不同内容(包括中文、英文、表格、图片、恶意代码等)的马尔可夫链熵率, 并计算嵌入恶意代码的文档熵率, 将待测文档熵率的区域传入布隆过滤器, 并与事先在宏代码中提取出来的关键字进行匹配, 判定文档性质。由于良性和恶意文档的连续字节会有重复的部分, 而且该方法仅仅根据关键字匹配度进行判定, 实验的准确率和误报率都会受到影响。文献[27]指出这种基于 n-gram 的检测方法对图片等文件检测效果较好, 但对 PDF 和 Office 文档的误报率很高, 因为这种检测方法实质上是对文件中是否嵌入对象进行了检测, 与图片等文件不同, PDF 和 Office 文档支持对象嵌入, 导致嵌入了对象的文档易被误判为恶意文档, 因而检测的误报率偏高。

文献[28]发现以前的统计分析模型存在一些不足, 比如没有考虑到一些恶意代码虽然很短, 但是却对文档性质判断起到很重要的作用。比如 Shellcode 代码, 提取了训练样本中信息增益最大的 500 个 3-gram 作为属性特征并训练检测模型, 实验结果表明提升了模型的性能。但是这类方法没有充分结合文档自身的特点, 不适用于恶意文档的检测, 因此, 出现了基于文档内容特征和结构特征的检测方法。

#### 4.1.2 基于内容特征的检测方法

苏璞睿等[29]提出了基于空间向量计算的方法, 这种检测方法的原理是提取文档中疑似 Shellcode 的数据片段与已知的 Shellcode 进行相似度计算, 从而判定文档的恶意性。这种检测方法有一定的局限性, 比如如何定位 Shellcode 入口点并提取 Shellcode 片段, 如何处理 Shellcode 混淆等, 并且这种检测方法的检测准确率受限于已知的 Shellcode 的规模, 并难以对新型攻击做出有效的检测。

2011 年, Laskov 等[30]提出了 PJScan, 这是第一个基于 JavaScript 对恶意 PDF 文档进行静态检测的模型。该模型使用 Poppler 对嵌入 PDF 文档中的 JavaScript 代码进行了定位和提取, 使用开源的 SpiderMonkey 引擎, 通过提取该引擎在解析 JavaScript 过程中生成的中间语言表示——词法标记

(Lexical Tokens), 并利用这种中间语言作为JavaScript 代码的特征描述, 由此构建 OCSVM(One-Class Support Vector Machine)分类模型对 PDF 文档进行分类。这种检测方法解决了 JavaScript 代码定位与提取的问题, 使用中间语言代替 JavaScript 能够更好的提取和表达 JavaScript 代码特征, 在一定程度上减少了混淆对特征提取的干扰。但是这种方法仍存在很多不足: 首先, 存在部分 PDF 攻击方式不依赖于 JavaScript 代码, 该模型不能检测这类攻击。其次, 该模型对代码混淆的处理效果仍然十分有限。

2012 年, Maiorca 等[31]通过研究文档的内容以及其中出现的关键词, 将这些文档内容和关键词作为文档特征, 并根据这些文档特征进行分类判定, 建立了 PDFMS 检测模型。这种检测方法实际上是选取了 PDF 文档中的元数据(Metadata)作为特征, 然后运用分类算法进行分类。这种检测方法突破了之前检测方法针对 Shellcode 和 JavaScript 代码检测的局限性, 并且实验结果显示这种检测方法取得了很好的检测效果。但是, 该方法对于一些元数据特征的选取与处理上还存在不足, 在处理恶意文档和正常文档的共有特征时, 只是通过简单的比较出现的次数进行区分, 这就造成攻击者在掌握判定标准时可以很容易的避免某些特征出现的次数, 从而实现攻击的隐藏, 这种基于检测特征恶意隐藏的攻击方法称为模仿攻击(Mimicry Attacks) [32]。

2013 年, 文献[33]提出了一种较为通用的多视角的恶意文档检测方法。在恶意文档中, 攻击者经常会利用习惯性的用词构筑触发漏洞的工具, 并且利用其中的功能关键字来触发恶意的大量的重复字段。所以, 作者提出了基于文档内容的三个特征, 分别是功能关键字, 习惯性用词和常数段对恶意文档进行分类。并提出了利用 TF-IDF (Term Frequency–Inverse Document Frequency, TF-IDF) 来标准化特征向量, 提高模仿攻击的检测率, 该方法的关注点在于提取文档内容特征, 通用性较强, 方法较为简单, 但是对于混淆文档没有很好的解决办法。文献[34-38]从恶意代码的识别、检测和反混淆等方面开展了相关研究, 也取得了一定成果。

#### 4.1.3 基于结构特征的检测方法

2012 年, Smutz 等[39]从 PDF 文档中提取了元数据和文档结构特征, 由此建立了一个更为精确的分类器, 该方法能够有效的对抗模仿攻击, 不过也出现了针对 PDF 文档结构特征检测的绕过方法[40]。2013 年, Šrndic 等[41]提出了一种基于 PDF 文档结构路径(Structural-Path)的特征表示方法, 这种检测方法通过提取 PDF 文档结构的层次路径信息进行建模, 利用决策树和SVM达到检测恶意文档的目的。Smutz 和 Šrndic 的检测方法都是通过更加深入地分析 PDF 文档的结构组成, 丰富文档判别的特征。不同的是, Šrndic 通过分析 PDF 文档的逻辑结构, 使提取的特征能够对文档提供更好的描述。

2015 年, Maiorca 等人[42]采用了提取元数据和文档结构特征对 PDF 文档进行分类判断的方法。与之前 Smutz 和 Šrndic 的检测方法不同, Maiorca 等人在提取文档结构时, 统计某些特征出现的次数; 同时, 在提取文档的元数据特征时, 考虑了嵌入恶意文件的攻击情况, 并将相应的特征加入到分类判定中。他们的方法扩展了恶意文档的检测范围, 增加了对嵌入恶意文件的检查。但是, 他们的方法也存在一些不足, 比如在文档结构特征的表示上比较简单, 这种简单描述方法不能很好的反映文档中资源的分布情况。

2016 年, Cohen 等[16]提出了 SFEM 方法, 这种方法借鉴了 Šrndic 等人提出的文档结构路径的特征提取, 并将这种方法应用到对 Office 文档的检测中去, 实验结果表明这种方法具有很高的准确率。同年, Šrndic 等[43]改进了他们之前的检测方法, 提出了抽象的思想, 提取不同文件的元数据特征并使用结构路径方法描述文档的结构特征, 再利用分类方法进行判定, 通过这种方法实现对 PDF、SWF 和 Office 等多种文件的静态检测, 并由此建立了新的检测系统 Hidost, 但是, 这种方法也存在很多需要改进的地方, 如对特征的抽象还不够深入, 对元数据特征的提取也不够深入等。

2017 年, 文献[44]提出了一种基于主动学习的恶意代码检测方法, 新型恶意代码大量涌现,其出现之初,样本数量有限,现有方法无法迅速检测出新型恶意代码及其变种。Nissim 等人[45]又在 SFEM 的基础上提出了基于主动学习的 Word 文档检测框架 ALDOCX, 该方法解决了之前检测方案需要大量学习样本以及人工干预的问题, 本文还提出了倾向于恶意文档样本的自主学习和与 SVM 简单边际学习相结合, 以符合不同数据集和不同检测需求, 但是该方法仅能检测 DOCX 文档, 对于文档内嵌入恶意链接的攻击无能为力, 所以仅仅基于结构特征分析的检测方法很容易被模仿攻击, 造成检测方式失效。

#### 4.1.4 基于多层抽象的检测方法

恶意文档的检测已经取得了许多成果, 但还存在恶意代码提取困难、难以应对模仿攻击以及模型泛化能力不足等问题。针对上述困难, 本文作者提出

了一种面向不同文档类型的鲁棒性强、可靠的特征提取方法[46], 设计了基于多层抽象的恶意文档检测方法[47-48], 基于抽象的思想, 研究不同类型文档的统一表达形式, 实现对多种文档的统一检测。检测方法的具体实现流程如图 8 所示。

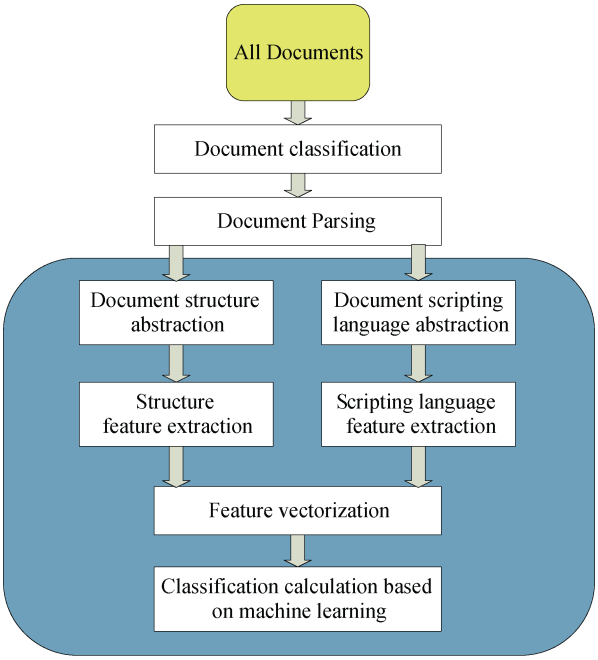


**图 8　检测算法流程图**

**Figure 8　A workflow of detection algorithm**

(1)文档分类及解析: 首先对文档进行分类, 去除格式不正确、无法打开等无效文档, 然后对文档进行解析, 分析文档的结构、内容等元素。

(2)文档结构抽象: 根据对 PDF、Word 等文档结构的分析可知, 不同文档是按照各自逻辑结构进行解析的, 这些文档的逻辑结构可视为一种树形结构, 对该结构进行抽象后得到统一的表示, 对每种节点提取了其节点的属性-值列表作为特征, 进行向量化计算, 如图 9 所示。

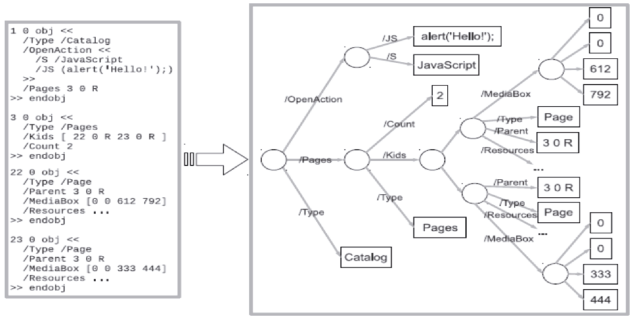


**图 9　文档结构抽象**

**Figure 9　Document structure abstraction**

(3)脚本语言抽象: 脚本语言是文档内容的重要特征, 基于文献[30]的词法标记方法, 设计了一种针对 JavaScript 和 VBA 脚本的中间标记语言, 实现对脚本语言的抽象实质是对文档内容特征的统一表达, 部分中间语言对照如表 2 所示。

**表 2　中间语言对照表(部分)**

**Table 2　Intermediate language comparison table (partial)**

| 符号名称 | 取值类型 | 描述 |
|---|---|---|
| Function | 缺省 | 函数定义的起始关键字 |
| Function_Num | Zero、Few、Multi | 自定义函数的数量 |
| Annotation_Num | Zero、Few、Multi | 代码中的注释数量 |
| AutoOpen_Num | Zero、Few、Multi | 自动执行函数数量 |
| Date_Num | Zero、Few、Multi | 时间函数数量 |
| Str_Num | Zero、Few、Multi | String 关键字数量 |
| For_Num | Zero、Few、Multi | 循环数量 |
| WebSite_Num | Zeor、Few、Multi | 脚本代码中网址数 |
| Func_In_Str | Zero、Few、Multi | 字符串中有函数名 |
| Str_Split | Zero、Few、Multi | 字符串分割操作 |
| Str_Replace | Zero、Few、Multi | 字符串替换操作 |
| Str_Mid | Zero、Few、Multi | 字符截取操作 |
| Str_Reverse | Zero、Few、Multi | 字符串倒置操作 |
| Application | Zero、Few、Multi | 关键函数, 数值表示频次 |
| Utilprintf | Zero、Few、Multi | 关键函数, 数值表示频次 |

(4)在完成文档结构和脚本语言抽象提取后, 对提取的数据进行向量化, 便于量化计算。

静态检测方法不实际执行样本, 因此具有检测安全性高、检测速度快、检测开销小等优点。但是也受到了一些质疑, 当前这种基于抽象的统一检测方法只是将文档的结构路径作为文档恶意性的判定依据, 而忽略了对文档中可能存在的恶意代码的特征提取。虽然基于结构路径抽象的检测方法效果很好, 但是也有相关论文[49-50]指出: 可以通过在恶意文档中添加正常文档的内容作为冗余数据, 使得恶意文档在特征的统计学分布上更接近于正常文档的方式绕过这种检测方法, 这种绕过检测的攻击方法属于模仿攻击。当前这种基于抽象实现对多种文档的检测方法, 在特征选取上避开了对嵌入的代码的分析和特征提取, 从而避免了对不同文档支持的不同脚本代码的解析, 这种方法虽然在一定程度上提高了该检测方法的普适性, 而且对常见的大多数恶意文档具有很高的检测准确性, 但是也造成了所选取的特征都是表象特征, 没有触及恶意文档的本质特征。由此, 这种方法对模仿攻击的检测能力不足。

## 4.2 动态检测方法

静态检测方法对于代码混淆等逃避检测技术一直没有好的解决方案, 动态调试可以通过在虚拟环境中打开文档, 监测文档运行过程中的行为来判定是否属于恶意文档。按照分析原理的不同, 动态检测方法根据动态仿真模拟的对象可以分为文档仿真法和代码模拟法。

### 4.2.1 文档仿真法

文档仿真法是指让文档在沙箱或虚拟机等虚拟环境中仿真运行, 最早由 Toth 等人[51]提出, 比较文档仿真运行前后状态参数的变化, 判定文档是否包含恶意代码, 监控的行为包括: 文件操作、模块加载、网络通信操作、注册表操作等。2007 年, Willems 等[52]提出了 CWSandbox 模型, 通过记录和检测样本在实际运行时的行为特征判定样本的恶意性, 该技术绕过了恶意文档静态检测方法所需要的复杂的恶意代码提取和还原工作, 对于经过混淆处理的文档也有很好的检测效果, 但是其时间和内存开销较大。

同年, 文献[25]在动态分析中, 结合三个动态行为变化对恶意文档进行判定: 首先作者认为 Word 文档在打开时需要加载大量的 DLL, 正常的文档是不会因为 DLL 加载顺序的改变而改变行为的, 于是作者在动态执行中改变 DLL 的加载顺序, 如果程序崩溃, 则证明是恶意的; 接下来, 作者将未被标识为恶意的样本用阅读器打开, 观察事先约定注册表的值是否发生变化, 如果发生变化则证明是恶意的; 如果上述两种情况都没有出现, 则观察文档打开时是否出现了已知恶意的弹窗, 如果出现则视为恶意文档。此种方法原理较为简单, 但是由于动态特征不全面而且不够有代表性, 导致误报率较高。随后, 文献[53]又提出了主动更改 Word 文档内数据部分的字节值并记录系统故障的方法作为之前动态分析的附加测试程序, 该方法主要针对将恶意代码嵌入在文档填充区域和文本数据部分的攻击。作者将处理后的文档运行在虚拟机中, 如果运行崩溃, 则证明存在恶意代码; 而正常文档则可以正常打开, 仅仅会发生字体、风格等变化。该方法可以抵抗模仿攻击和穷举攻击, 但此种方法需要深入理解 DOC 文档的结构, 并且对宏病毒等攻击无效。

2017 年, 文献[54]提出不同操作系统用相同的阅读器打开 PDF 文档, 然后追踪内部 PDF 进程, 从而观察不同操作系统对文档的影响。经过研究发现, 如果文档是良性的, 那么文档在某些维度上表现是相同的, 但是恶意文档会表现出不同的行为。

### 4.2.2 代码模拟法

代码模拟法是指针对恶意文档内嵌的 Shellcode、JavaScript 等代码进行模拟执行。最开始用于网络数据流 Shellcode 的检测。比如基于硬件化模拟器 ShellOS[55]、基于 CPU 仿真的 Libemu 和 QEMU 模拟器[56]以及现在比较流行的基于软件的仿真技术。2012 年, Schreck 等[57]提出了一个可以自动确定 Word 文档所利用漏洞攻击的方法。该方法分为三个步骤: (1)区分良性文档和恶意文档; (2)提取嵌入的恶意 Shellcode 代码; (3)确定攻击利用了 Word 文档的哪种漏洞, 漏洞分为已存在补丁的漏洞和新型漏洞。论文弥补了对文档漏洞攻击检测方面不足, 但仅能检测 DOC 文档, 且检测准确率取决于 Shellcode 提取的准确性。文献[58-60]针对恶意 JavaScript 脚本代码的检测提出了动态追踪等反混淆方法。

2016 年, Iwamoto 等人[61]提出了一种基于字节序列熵和仿真的方法来提取恶意文档中的 Shellcode。作者首先对文档进行反汇编并计算字节序列熵, 将熵值较大的字节序列作为 Shellcode 候选集合, 并生成一个可执行文件。将可执行文件进行模拟, 如果发现可执行文件具有自修改代码, 访问 PEB 和调用系统 API 的现象就将此字节序列定义成 Shellcode。此方法解决了动态触发漏洞较难的问题, 但是这种方法有一定的局限性, 比如对那些需要特定环境和初始值的 Shellcode 无能为力。本文研究者针对恶意 Word 中 Powershell 的提取和检测问题, 提出了两层解混淆的方法, 实现了对恶意 Powershell 更加准确的检测效果[62]。

动态检测方法的优势在于不用对样本进行学习, 可直观的发现攻击者攻击行为及其攻击目的, 具有很强的健壮性, 但是动态检测方法通常会消耗大量资源和内存空间, 而且会受到反虚拟机技术的对抗。不同于对 PE 等可执行文件的检测, 纯动态检测方法对恶意文档检测存在很多局限性。由于动态检测对触发的要求较高, 其中有很大一部分恶意文档往往需要在执行前诱导用户修改文档的安全设置, 诱导用户点击、确认等行为才能触发恶意代码执行。同时, 不同版本的文档阅读器存在的漏洞也不相同, 而针对某一漏洞的恶意代码只有在特定版本的阅读软件中才能执行, 这些问题都极大的限制了恶意代码执行的触发成功率, 从而使得动态检测监控不到相应的恶意行为。

## 4.3 动静态结合检测方法

研究人员逐渐开始将文档在打开时的行为特征与文档自身的特征结合起来进行分析, 由此产生了

动静结合的恶意文档检测方法。根据检测目的的不同, 可以分为独立特征检测法和特征融合检测法。

#### 4.3.1 独立特征检测法

独立特征检测法是指通过动态执行解决文档中恶意代码的去混淆问题, 再将去混淆后的代码使用静态检测分析方法判定恶意性。2011 年, Tzermias 等[1]提出了动静态结合的文档检测模型 MDScan, 这个模型通过提取嵌入 PDF 中的 JavaScript 代码, 并将代码送入 SpiderMonkey 中, 在仿真环境下执行, 从而提取可能存在的 Shellcode。然后, 对疑似为 Shellcode 的数据片段输入 Nemu[63]进行判断。这种检测方法虽然能够利用动态检测的优势, 有效的消除代码混淆的影响, 但是它与针对 JavaScript 的静态检测方法一样存在难以解决 JavaScript 代码定位与提取的问题, 同时也存在只能对针对基于 JavaScript 攻击方式进行检测的局限性, 而且检测的准确率受限于静态检测 Nemu 的检测效果, 无法检测新型攻击。文献[64-65]通过解析文档提取 JavaScript 代码, 经 SpiderMonkey 分析器提取静态行为, 再利用 YARA 编写规则库进行恶意代码识别, 获得静态扫描报告。在动态分析部分, 将静态检测提取出的 JavaScript 代码输入 DIStorm 等反汇编工具, 通过人工分析提取出 Shellcode 代码, 再将 Shellcode 代码输入 Libemu 进行检测。这种方法由于依赖人工分析来提取 Shellcode, 自动化能力不足, 无法对大规模的恶意文档进行检测, 并且检测的结果受限于 YARA 规则库的完备性和 Libemu 的检测效果。

2013 年, 诸葛建伟等[66]提出了 MPScan 检测模型, 与 Tzermias 等人的思路类似, 其独到之处在于该方法直接对 Adobe Reader 进行 Hook, 通过本地 Adobe Reader 执行 JavaScript 代码提取出疑似的 Shellcode, 然后输入 Libemu 进行判断。这种方法解决了对 JavaScript 代码的提取和去混淆问题, 但是仍然不能解决检测的局限性问题, 即对不基于 JavaScript 的攻击方法没有检测能力, 同时检测的准确性受到 Libemu 的限制, 无法检测新型攻击。

文献[67-69]利用动态方法提取恶意代码, 然后与静态检测方法结合, 判定文档恶意性。这类方法利用动态仿真提取的恶意代码作为静态检测方法的输入, 但是仍然存在恶意代码触发成功率低、资源消耗大等问题, 检测方法适用范围较小。类似的工作还包括 CUJO[70]、ZOZZLE[71]、Lux0R[72]等。

#### 4.3.2 融合特征检测法

特征融合法是指为了将动态行为特征与静态特征结合起来作为判定文档恶意性的依据, 扩大了特征选取的范围。2014 年, 文献[73]提出了结合文档元数据特征和动态行为特征的检测方法。这种检测方法从文档的元数据中提取了少量特征, 监控 PDF 文档阅读器在打开文档时的行为特征, 然后结合两者的特征进行文档的判断。这种检测方法加入了文档元数据特征的提取, 同时改变了通过动态执行 JavaScript 提取可疑的 Shellcode 数据片段的传统思路。在动态检测时采用的方法是监控阅读器中 JavaScript 代码的执行, 根据这些行为特征与之前的元数据特征进行统一判定。这种检测方法的好处是引入文档的元数据特征使得特征表述更为详细, 对提高检测的准确性有很大帮助; 同时, 进程行为的检测避免了对 JavaScript 进行定位和去混淆处理, 通过对行为的监控摆脱了 Shellcode 特征库的限制, 使得这种方法能够对部分新型攻击进行检测。

2017 年, 文伟平等[74]针对漏洞 CVE-2011-2096、CVE-2011-2097、CVE-2011-2098 的具体攻击利用, 构建了静态匹配规则。针对恶意 PDF 文档中的 ROP 链进行研究, 通过 Hook 敏感 API 函数对堆栈、内存、动态链接库的实际布局与状态的关键标记特征进行监控, 从而判断是否存在实现攻击的 ROP 链。这种检测方法不仅能够实现对文档的恶意性判定还有识别攻击原理及具体漏洞利用的能力。但是, 该方法匹配规则需要人为设计与添加, 存在系统检测的攻击种类受限, 扩展与更新依赖于经验等不足。

总体而言, 可以将动静态结合的文档检测方法分为两类: 一类是通过动态执行解决文档中恶意代码的去混淆问题, 再将去混淆后的代码使用静态检测分析方法判定恶意性; 第二类是将动态行为特征与静态特征结合起来作为判定文档恶意性的依据, 扩大了特征选取的范围。理论上该方法应该比静态检测方法更为有效。但是, 该方法仍不能避免动态检测中恶意代码触发成功率低的问题, 以及相比较于静态文档检测仍存在资源成本消耗大, 速度慢等不足, 同时, 虽然是在虚拟环境中运行样本, 但相比静态检测仍具有更高的风险, 因此这种方法不适宜在终端进行检测, 而更适合应用在分布式集群中。

### 4.4 其他相关研究

随着机器学习、深度学习在恶意软件检测中的应用[75-76], 对抗攻击开始出现, 早在 2004 年, Dalvi 等[77]就对对抗攻击做了研究, 他们尝试构造逃避策略对基于统计的垃圾邮件过滤器和恶意软件检测器进行逃避, 并提出系统化的针对线性分类器的攻击方法, 基本思想为篡改垃圾邮件的内容。Šrndic 等[78]实现通过简单梯度下降攻击算法来规避非线性

SVM和神经网络, 报告了针对非线性学习算法的第一个对抗样本, 而这也是第一个展示出在黑盒攻击场景下对抗样本的转移性的实验, 将对抗攻击扩展到多类分类器。2016 年, 文献[79]研究了一种更有约束的逃避攻击场景, 攻击者只有对目标检测器的黑盒访问权, 目标检测器会为攻击者的输入样本输出实值分类得分, 使得可以通过遗传算法生成能够实现逃避攻击的对抗样本。2017 年, 文献[80]研究了一个更现实的黑盒攻击场景, 目标分类器与攻击者的接触最少, 只显示最终的分类决策(拒绝或接受输入样本), 而且攻击者只能使用黑盒变形操纵恶意样本, 并提出了一种基于有限信息的计分机制, 它可以根据每个样本的逃避进度, 给出一个真实值的计分方法。

与前文提到的模仿攻击不同, 对抗攻击主要表现为从分类器使用的算法、特征集或训练数据出发, 精心构造对抗样本, 使得分类器在测试时输出错误的结果。例如, 攻击者可能会对一个恶意样本结构进行变形, 使它重新被分类为类似训练集中的一个良性实例, 从而逃避目标分类器[81-82]; 也可以改变恶意样本的一些特征, 使分类算法决策失误, 从而逃避目标分类器[83-84]。

恶意文档的漏洞挖掘、利用等技术研究对恶意文档的检测有重要借鉴与启发意义, 在漏洞挖掘方法上有以文献[85-87]为代表的通过 Fuzzing、符号执行、反汇编和虚拟执行等技术实现恶意文档的漏洞挖掘。在实现漏洞利用攻击与逃避检测方面的研究有文献[88-90]等, 在他们的研究中详细介绍了实现攻击与躲避检测的原理, 并提出了相对应的检测思路。

以 Endignoux 等[91]为代表的学者提出对原文档进行规范化处理生成安全的标准文档的思路来防御恶意文档攻击的思路。2015 年, Smutzd 等[92]受到操作系统随机地址保护的启发, 在文件格式级别修改 Word 文档, 转换后的文档的功能和原来文档一样, 但却可以破坏恶意内容, 使其失去攻击行为。该方法难度较大而且仅能对部分针对文档漏洞的攻击有效。Mai 等[93]则专门对新出现的恶意 PDF 文档的特征进行了最新的统计。

本文将恶意文档的检测方法归纳为静态检测、动态检测、动静结合检测以及其他相关研究等四大类, 从方法名称、方法描述、代表性工作、适用场景、优点和缺点等角度详细地对现有检测方法进行总结和对比, 尤其是在适用场景中, 对如何合理、高效地使用每种方法给出了具体的建议, 增强了指导性, 具体内容如表 3。

## 5 性能评价

本节归纳并总结恶意文档检测的评价方法、数据集、检测工具、辅助分析工具以及在线检测平台等内容, 这些数据集和工具在研究中得到了广泛的应用, 能够支撑相关技术的研究。

### 5.1 评价方法

恶意文档检测的评价方法较多, 可以归纳为统计相关评价、准确性相关评价以及其他评价。

#### 5.1.1 统计相关评价

统计相关评价通常使用交叉验证方法(Cross Validation, CV), 是一种将数据样本统计切割成更小子集的实用方法。使用训练数据集训练参数时, 整个训练集通常分为三部分: 训练集、评估集和测试集, 这是专门为检验训练效果而设置的。其中, 测试集是完全不参与训练的数据, 仅用于观察测试数据, 在实际训练中, 训练结果通常在训练集上表现良好(初始测试条件是敏感的), 但对训练集之外数据效果欠佳。因此, 我们通常不训练所有数据集, 而是划分一部分(例如 10 倍)来测试训练集生成的参数, 这可以相对客观地判断训练集之外数据的表现。常见的交叉验证方法如下, 保持法(Hold-Out Method), $K$ 折交叉验证($K$-fold CV), 一次性交叉验证(Leave-One-Out CV)等。

(1)保持法: 通过将样本随机分成两个分离部分来进行统计评估, 比如 70%的样本用于训练系统, 而剩余部分用于评估其在新样本上的分类准确度。每个分割都包含由恶意和良性文档组成的样本, 重复上述过程 $N$ 次用来估计检测器精度的平均值和标准偏差。在实验期间评估阈值和分类不同的算法, 例如随机森林(Random-Forest), LogitBoost, 逻辑回归(Logistic Regression)和支持向量机(SVM)等。

(2)$K$折交叉验证 ($K$-CV): 原始数据被分成$K$组, 每个子集数据单独验证, 剩余的 $K$-1 子集数据被用作训练集, 从而获得并且可以使用 $K$ 个模型。验证集的分类精度平均值用作该 $K$-CV 下类别的性能指标。$K$一般大于等于2, $K$-CV能有效地避免过度学习和学习不足, 所得结果更有说服力。

(3)留一法交叉验证 (LOO-CV): 如果原始数据有 $N$ 个样本, 那么 LOO-CV 为 $N$-CV, 即每个样本单独用作验证集, 其余 $N$-1 样本作为训练集, 所以 LOO-CV 将得到 $N$ 个模型。这 $N$ 个模型的最终分类精度的平均值被用作 LOO-CV 分类的性能指标。LOO-CV有两个明显的优势: 首先, 几乎所有的样本都被用来训练模型, 因此它最接近原始样本的分布,

**表 3　恶意文档检测方法比较**

**Table 3　Comparisons of malicious document detection methods**

| 方法分类 | 方法名称 | 方法描述 | 代表性工作 | 适用场景 | 优点 | 缺点 |
|---|---|---|---|---|---|---|
| 静态检测 | 基于统计分析 | 提取字节序列作为特征, 利用 n-gram 等计算马氏距离、熵率等。 | Schultz[22] Stolfo[24-25] | 适用于文档中的恶意代码提取完整、准确的情形, 计算速度快。 | 不实际执行样本, 具有检测安全性高、检测速度快、检测开销小等优点。对文档结构、内容等抽象, 在特征选取上避免了对复杂文档支持的不同脚本代码解析, 提高了检测算法的普适性。 | 忽视了对文档中恶意代码的提取, 提取的抽象特征没有触及恶意文档的本质, 造成这类方法对模仿攻击的检测能力不足。 |
| | 基于内容特征 | 检测文档中的恶意代码, 比如 Shellcode、JavaScript, 或者分析文档关键词、习惯用词等元数据特征。 | Laskov[30] Maiorca[31] Chen[36] | 适用于文档中的恶意代码未采取混淆、加密等对抗检测技术, 检测结果准确率高。 | | |
| | 基于结构特征 | 提取文档逻辑结构的路径、层次等特征, 利用决策树、SVM 等机器学习方法进行分析。 | Maiorca[40-42] Šrndic[41-43] Nissim [45] | 对于没有采用对象嵌入、远程链接等攻击技术的文档具有较好的检测效果。 | | |
| | 基于多层抽象 | 对文档结构和脚本语言进行抽象、向量化, 利用机器学习方法进行计算。 | Yu[46-48] | 适用于大多数恶意文档的检测, 但是对于采用对抗机器学习等技术的恶意文档检测能力不足。 | | |
| 动态检测 | 文档仿真法 | 文档在沙箱或虚拟机等环境中仿真运行, 提取并分析文档打开、编辑等行为恶意性。 | Toth[51] Xu[54] | 适用于对文档内嵌入恶意代码类型无法提前知晓的情况, 通过仿真分析调用的资源。 | 不用对样本进行学习, 可直观的发现攻击者攻击行为及其攻击目的, 具有很强的健壮性。 | 消耗大量资源和内存空间, 而且会受到反虚拟机技术的对抗。由于文档动态检测对触发的要求较高, 监测恶意行为难度大。 |
| | 代码模拟法 | 对恶意文档中内嵌的 Shellcode、JavaScript 等脚本进行模拟执行, 动态追踪并分析脚本运行行为。 | ShellOS[56] Schreck[58] Iwamoto[62] | 对内嵌混淆或加密处理过的 Shellcode、JavaScript 脚本以及宏病毒文档具有较好的检测效果。 | | |
| 动静结合检测 | 独立特征法 | 通过动态执行解决文档中恶意代码的去混淆问题, 再将去混淆后的代码使用静态检测分析方法判定恶意性。 | Zhuge[67] CUJO[70] ZOZZLE[71] | 适用于已经提前知晓文档中的核心代码被混淆或加密等处理过, 利用动态方法提取完整的恶意代码进行静态分析。 | 通过动态分析一定程度上解决了代码混淆和模仿攻击的问题, 适合在分布式机器集群中进行检测。 | 不能避免动态检测方法的恶意代码触发成功率低的问题, 耗费资源多, 检测耗时, 不适宜在终端进行检测。 |
| | 融合特征法 | 为了将动态行为特征与静态特征结合起来作为判定文档恶意性的依据, 扩大了特征选取的范围。 | Liu[73] Wen[74] | 该方法用于希望扩大文档各类特征的情形, 提高了文档特征的维度, 加大的计算量。 | | |
| 其他相关研究 | 对抗攻击检测 | 从检测器分类使用的算法、特征集或训练数据出发, 精心构造出对抗样本, 导致分类结果的错误。 | Zhang[75] Šrndic[78] | 在知晓对抗分类器算法的攻击具有一定的检测能力, 但是泛化能力不足。 | 从对抗攻击、漏洞挖掘利用、文档规范等新的角度来讨论文档安全问题, 为恶意文档的检测提供了新思路。 | 具有较强的针对性, 无法实现对大规模文档的检测。 |
| | 漏洞挖掘与利用 | 从文档漏洞挖掘、利用等方面提出了恶意文档检测的新路径。 | Mcqueen[85] Mei[86] | 适用于具有自动化漏洞挖掘平台或系统的情形, 也可以作为恶意文档分析的重要组成部分。 | | |
| | 其他研究 | 从文档规范化、文档格式修改等独特角度讨论了恶意文档检测的新思路。 | Endignoux[91] Smutzd[92] Mai[93] | 定位了保护系统不受恶意文档攻击的目的, 实现“以防代检”。 | | |

得到的结果更可靠。第二, 实验中的数据不受随机因素的影响, 保证了实验过程的可复制性。然而, LOO-CV 的缺点是计算成本高, 因为要建立的模型的数量与原始数据样本的数量相同。因此, 当原始数据样本数量相当大时, LOO-CV 在实际应用中是非常困难的。

### 5.1.2　准确率相关评价

恶意文档检测最重要的评价指标是文档检测的准确性相关指标, 常用的指标有精确率(Accuracy)、准确率(Precision)、召回率(Recall)、真阳性率(True Positive Rate, TPR)、假阳性率(False Positive Rate, FPR)、真阴性率(True Negative Rate, TNR)、假阴性

率 (False Negative Rate, FNR) 、F-Score( 假 定 F1-Score)、受试者工作特征曲线(Receiver Operating Characteristic, ROC)、ROC 曲线下的面积(Area Under Curve, AUC)。其中, Accuracy 是对于给定的测试数据集, 分类器正确分类的样本数与总样本数之比, 可以理解为所有实验中, 分类正确样本的比例; Precision 表示恶意文档被检测为恶意文档数量占全部检测为恶意文档数量的比例, 可以理解为恶意文档被正确检测的比例, 恶意文档找对的比例; Recall 和 TPR 具有相同的公式和结果, 表示恶意文档被检测为恶意文档的数量占所有恶意文档数量的比例, 可以理解为恶意文档中有多少被检测成恶意文档, 恶意文档找全的比例; FPR 表示检测为恶意文档实际是良性文档的个数占总体中所有良性文档的比例, 可以理解为良性文档例被误报的比例; TNR 表示所有良性文档中, 有多少被检测为良性样本; FNR 表示所有恶意文档中, 有多少被检测为良性文档; F-Score 是 Precision 和 Recall 的加权调和平均值, 综合了两者的结果, 一般取 F1-Score, 当该值较高时, 说明结果较理想; ROC 可以反映二分类器的总体分类性能, 当测试集中的样本分布发生变化的时候, ROC 曲线能够保持不变, 因此能够更客观的进行分类器性能的评价, ROC 对应的 AUC 越大(或者说对于连续凸函数的 ROC 曲线越接近(0,1) ), 说明分类性能越好。

我们首先定义四个变量, 这也是机器学习中常用的定义, 如表 4 所示。真阳性(*True Positive, TP*): 检测结果为良性文档, 实际上就是良性文档; 假阳性(*False Positive, FP*): 检测结果为良性文档, 实际上是恶意文档; 假阴性(*False Negative, FN*): 检测结果为恶意文档, 实际上是良性文档; 真阴性(*True Negative, TN*): 检测结果为恶意文档, 实际上就是恶意文档。公式(1)~(8)为相关评价指标的计算方法。

**表 4 变量定义表**

**Table 4 Variable definition**

| | 检测为恶意 | 检测为良性 |
|---|---|---|
| 恶意文档 | *TP* | *FN* |
| 良性文档 | *FP* | *TN* |

$$\text{Acurracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{TPR} = \frac{TP}{TP + TN} \tag{4}$$

$$\text{FPR} = \frac{FP}{TN + FP} \tag{5}$$

$$\text{TNR} = \frac{TN}{FP + TN} \tag{6}$$

$$\text{FNR} = \frac{FN}{FP + TN} \tag{7}$$

$$\text{F1} - \text{Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{8}$$

### 5.1.3 其他评价

除了上述评价指标外, 还有如下评价方法:

(1)基于 CVE 的评估: 识别由已知漏洞利用的恶意软件样本组成的子集, 通过将子集分成两个分离部分来执行基于 CVE 的评估。第一部分用于训练系统, 它由 PDF 恶意软件组成, 利用在某一年 *Y* 之前发现的漏洞, 剩余部分用于评估系统对前所未见的漏洞(即 *Y* 年后发现的漏洞)的检测准确性。因此, 学习良性样本作为统计评价, 评价其假阳性率, 平均检测结果需要运行 *N* 次。

(2)对抗性评估: 根据检测系统的体系结构, 设计不同抽象层次的对抗样本, 然后测试系统对抗恶意文档样本的对抗操作。例如, 我们可以在设计恶意 PDF 样本时在已公开的漏洞中添加 API 引用, 目的是逃避系统的检测。

## 5.2 数据集

方法的训练、评估和验证都需要依赖普遍、有代表性的数据集, 恶意文档数据集的来源通常包括: 一是知名恶意软件厂商, 如 VirusTotal、VirusShare 等; 二是相关论坛、博客等分享, 如 Contagio; 三是论文中提供的数据集, 如 Srndic 和 Laskov 等, 目前已收集数据如下表所示。

**表 5 公开样本数据集**

**Table 5 Public sample data set**

| 数据集来源 | 年 | 恶意文档数量 | 良性文档数量 |
|---|---|---|---|
| VirusTotal | 2010-2015 | 17,923 | 15,517 |
| Contagio | 2010 | 410 | 100 |
| Srndic & Laskov | 2012-2017 | 27,757 | — |
| 其他 | 2013-2019 | — | 6,012 |
| 总计 | | 46,090 | 21,629 |

从上表可以看出, 良性样本对于恶意样本是欠采样的, 针对样本不均衡问题, 研究人员也正在不断的构建更好的数据集, 包括本文作者也构建一个针对 PDF 和 Word 文档的数据集[47-48], 如下表所示:

表 6 自建数据集

Table 6 Private sample data set

| | 恶意 PDF | 良性 PDF | 恶意 Word | 良性 Word |
|---|---|---|---|---|
| 数量 | 12,000 | 20,000 | 1000 | 2000 |
| 大小 | 7.72MB | 8.67MB | 603KB | 856KB |

## 5.3 工具和平台

近年来, 恶意文档自动化检测的需求越来越旺盛, 这对恶意文档检测工具、辅助分析工具及在线检测平台的发展起到了重要的推动作用, 代表性的工具及平台如下。

### 5.3.1 检测工具

- PJScan[94]: 是一个命令行应用程序, 它使用机器学习算法来检测嵌入 JavaScript 相关的恶意 PDF 文档, PJScan 这个名字是“PDF 和 JavaScript Scanner”的首字母缩写。
- JSunpack-n[95]: 一种命令行形式的 JavaScript 解析器, 专为自动检查和反混淆 JavaScript 而设计, 附带了一个 pdf.py 脚本, 该脚本能够解压缩 PDF 文档中包含的 JavaScript, 可以检测 PDF 文档是否存在恶意 JavaScript 文件。
- OfficeMalScanner[96]: 一个用于扫描恶意跟踪的 Office 取证工具, 如 Shellcode、PE 文件或嵌入式 OLE 流, 它支持反汇编和 hexview 以及简单的强力模式来检测加密文件, 也能对文档的 VB 宏代码进行提取, 以供进一步分析。类似的还有面向 RTF 文件的 RTFScan。
- QuickSand[97]: 用于分析可疑的恶意文档, 可以定位和提取嵌入文档的可执行文件, 还具备检测包含 0day 或未知混淆漏洞文档的能力, 可作为命令行工具运行, 也能集成到其他产品中。

### 5.3.2 辅助分析工具

- Libemu[98]: 一款用 C 语言实现的基于 x86 的 Shellcode 辅助分析库, 可以实现静态分析、动态分析、Win32API hook 等功能, 可以判断字符串是不是 Shellcode, 或者用 Libemu 得到指令执行流程图(类似于 IDA 等调试工具)。
- SpiderMonkey[99], ViperMonkey[100]: 用于分析和反混淆文档中包含的恶意 JavaScript 或 VBA。
- Peepdf[101]: 一种允许用户浏览和分析 PDF 文档内容的交互式命令行工具, 它可以解析文档, 显示文档结构, 并列出概要信息、元数据、错误信息等。还可以提取 JPDF 嵌入的 avaScript 脚本进行解码、反转义、执行等各种操作。
- Origami PDF[102]: 是一个用于操作 PDF 的 Ruby 框架, 具有 PDF 解析器, 可以分析、修改、创建恶意的 PDF 文档, 支持 JavaScript、附件、表单等 PDF 规范。
- Offvis[103]: 它将显示 DOC, XLS 或 PPT 文件的二进制结构, 并确定一些常见漏洞。
- OLETools[104]: 基于 OLEfile 解析器, 用于分析 Office 和 OLE 文件安全特性的 Python 工具, 主要用于恶意软件分析, 取证和调试。
- YARA[105]: YARA 是一个旨在帮助恶意软件研究人员识别和分类恶意软件样本的工具。使用 YARA, 可以基于文本或二进制模式创建恶意软件系列的描述, 每个描述, 由一组字符串和一个确定其逻辑的布尔表达式组成。

### 5.3.3 在线检测平台

- ThreatAnalyzer (CWSandbox) [106]: 提供快速的 PDF、Office、木马等各类恶意软件样本的分析, 通过执行内部控制环境下的代码和恶意软件分析系统(沙箱), 可安全地运行恶意文件或分析恶意软件的行为方式。
- PDF Examiner[107]: 支持用户浏览文档结构, 以及检查、解码和转储 PDF 对象内容, 支持基本 PDF JavaScript 混淆, 加密 PDF(RSA, AESV2, AESV3), 剖析 PDF 流以发现新的和已知的漏洞, 适合手工的 PDF 分析任务。
- ViCheck.ca[108]: 提供对高级恶意软件检测引擎的访问, 支持解密和提取常见文档格式(如 Word, Excel, PDF)的恶意可执行文件。
- VirusShare[109], VirusTotal[110]和 VirSCAN[111]: 知名的多引擎恶意代码在线检测平台, 可以分析包括恶意文档在内的各种恶意文件, 并给出检测报告。

## 6 结论

本文系统概括分析了恶意文档检测的方法、技术和工具平台, 这些恶意文档通常作为实施 APT 攻击的重要一步, 因此对恶意文档的检测和分析方法已经成为近年来研究的热点, 有许多方面的工作值得深入研究下去。具体来说, 主要包括以下几个方面:

(1) 恶意文档的统一检测模型。随着国产化替代工程的深入, 以 WPS、OFD 等为代表的文档格式将在我国部委等机关单位占据重要的地位, 这些文档与 Office、PDF 等文档在结构、内容等方面均存在较大差异, 如何建立统一的检测模型, 还存在以下方面的问题: 一是在深入分析文档组成的同时, 特征选取的范围逐渐扩大, 从 Shellcode 与脚本代码的特

征选取, 到对所有元数据的选取, 再到整个文档结构特征的选取, 使得特征选取尽可能全面的描述文档的属性; 二是从对文档整体进行检测到将文档分模块检测, 再到细致的提取每个模块内的不同属性, 后来又发展到对文档的结构特征进行提取, 使得选取的特征具有更加准确与突出的描述能力; 三是提升深度学习在安全检测方面已表现出比较好的性能, 但在恶意文档检测领域还没有应用。四是如何挖掘这些文档的共同特征, 建立统一的检测模型。

(2) 真实恶意代码的提取。恶意文档的检测本质上还是要分析发现文档中存在的恶意代码, 因此如何有效、准确地提取文档中嵌入的真实恶意代码也是今后工作的一个难点。一是当前多数检测方法所使用的文档阅读器, 设计初衷是实现对文档的读写等正常操作, 为实现软件稳定性往往增加了容错机制, 导致了安全性和规范的审查与执行上存在逻辑漏洞, 攻击者能够利用文档阅读器存在的逻辑漏洞, 使得恶意代码无法被阅读器提取出来; 二是代码混淆使得执行代码的真实意图被刻意隐藏, 或者使得恶意代码片段无法被完整提取, 从而造成恶意代码提取困难, 或者提取出的恶意代码不准确, 导致无法检测; 三是文档中的恶意性特征, 比如结构、属性关键字等, 这类特征的提取采用的是关键字匹配或者正则表达式的方式进行字符的匹配, 攻击者为了躲避检测, 会对这类特征进行混淆处理, 使得处理后的内容无法通过原有的字符串匹配的方法获得, 已有方法目前只能对已知的混淆方法进行解混淆, 适用性远远不足。

(3) 对抗检测技术研究。随着恶意文档技术的发展, 对抗检测的文档也开始出现, 一是当前人工智能发展迅猛, 尤其是机器学习受到了广泛的应用, 然而, 对抗机器学习的恶意样本已经出现, 导致人工智能所驱动的识别系统出现混乱, 形成漏判或者误判, 甚至导致系统崩溃或被劫持, 并可以使智能设备变成僵尸攻击工具, 目前, 这类恶意文档检测的研究还有很大的提升空间; 二是模仿攻击, 攻击者可以在已经知晓检测特征后故意隐藏特征, 或者在保证攻击实施的基础上, 大量添加正常文档所具有的特征来干扰分类器的判定结果躲避检测。PDF 文档中实际数据是以对象为单位进行存储, 对象之间通过相互引用最终构成 PDF 阅读时的页面。为了应对机器学习的判定方法, 部分恶意 PDF 文档采用添加正常文档的对象使得提取的特征所构成的向量更加接近于正常文档, 从而导致机器学习算法失效。这种添加正常文档对象的方法造成被检测的恶意样本在统计学规律表现上非常接近于正常文档, 从而会被错判为正常文档, 逃过检测。如何提高文档检测方法的健壮性、抗干扰能力是下一步研究难点。

(4) 文档漏洞分析。文档漏洞分析是文档安全的重要部分, 基于文档阅读器解析漏洞的攻击占据很大比重, 但如何准确提取 Shellcode、如何确定漏洞版本、如何发现新型漏洞仍然是一个难题。一是由于当前的检测方法大多是基于文档阅读器来解析文档, 而这类阅读器的设计初衷是实现对文档的读写等正常操作, 为了实现软件的稳定性, 这类软件往往增加了容错机制, 而这导致了在安全性和规范的审查与执行上存在逻辑漏洞, 因此, 攻击者能够在利用文档阅读器存在的逻辑漏洞, 使得恶意特征无法被阅读器提取出来, 从而实行躲避检测的目的; 二是新型漏洞的发现与利用使得利用恶意代码库进行的匹配的检测方法失效, 而应对 0day 漏洞攻击正是当前各种检测方法最为薄弱的地方; 三是目前对于恶意文档行为的破坏是基于动态的, 而且检测范围受限, 如何深入理解文档结构, 开发出一种高效的, 适用范围广的主动攻击技术, 也是很好的研究点。

(5) 文档夹带、伪装等问题。当前对文档的检测大多集中在对文档中嵌入的恶意代码的检测。而恶意文档的攻击方式还有嵌入恶意文件(如图片、文档、可执行文件等), 或在文档中嵌入恶意链接等方法。当前, 无论是静态检测还是动态检测, 对于这类攻击的研究都很少。而这部分工作除了对文档进行分析检测外, 还需要增加对图片隐藏攻击、可执行文件分析、链接内容检测等方面的分析工作, 加大了文档检测工作的难度。

(6) 复杂文档分析技术。当前的恶意文档研究存在“重检测、轻分析”的问题: 恶意攻击越来越趋向于多种攻击技术融合, 共享威胁情报可以加速攻击事件的响应效率。没有进一步对该文档进行深入分析, 挖掘并关联分析得到其他可能存在的更多恶意特征, 为后续的恶意文档解析、特征统一提取、文档统一检测等提供更多的思路, 同时新型漏洞的发现与利用使得利用恶意代码库进行匹配的检测方法失效, 而应对 0day 漏洞攻击正是当前各种检测方法最为薄弱的地方。恶意文档攻击呈现大规模撒网式攻击, 对恶意文档类型进行分类并进行关联分析, 对于溯源取证很有帮助。

总之, 恶意文档检测的研究虽然存在一些问题, 但是这些研究问题也为科研人员提供了重要的研究基础和工程价值, 也为恶意文档检测的未来研究方

向提供了重要的借鉴与启发意义。

**致　谢**　本文的工作得到了刘云政、夏彬、娄尘哲、李佳楠等同学的帮助, 在此表示诚挚的感谢。

## 参考文献

[1] Tzermias Z, Sykiotakis G, Polychronakis M, et al. Combining Static and Dynamic Analysis for the Detection of Malicious Documents[C]. *The Fourth European Workshop on System Security - EUROSEC '11*, 2011: 1-6.

[2] Garber L. Melissa Virus Creates a New Type of Threat[J]. *Computer*, 1999, 32(6): 16-19.

[3] How Just Opening an MS Word Doc Can Hijack Every File on Your System. The Hacker News. https://thehackernews.com/2016/02/locky-ransomware-decrypt.html. 2016.

[4] PoC Attack Leverages Microsoft Office and YouTube to Deliver Malware. Threatpost, https://threatpost.com/poc-attack-leverages-microsoft-Office-and-youtube-to-deliver-malware/138585/. 2018.

[5] Malware Reports: IT threat evolution Q3 2018. Kaspersky. https://securelist.com/it-threat-evolution-q3-2018-statistics/88689/. 2018.

[6] PDF Reference. Adobe. https://www.adobe.com/devnet/pdf/pdf_reference_archive.html. 2019.

[7] XML and Microsoft Office Word 2003: Writing a Trip Report. Microsoft. https://docs.microsoft.com/en-us/previous-versions/Office/developer/Office-2003/aa537158(v=Office.11). 2014.

[8] Overview of the XML file formats in Office 2010. Microsoft. https://docs.microsoft.com/en-us/previous-versions/Office/Office-2010/cc179190(v=Office.14). 2011.

[9] Rich Text Format (RTF) Specification, version 1.9.1. Microsoft Corporation. https://docs.microsoft.com/en-us/previous-versions/Office/developer/Office2000/aa140280(v=Office.10). 2014.

[10] WPS. https://www.wps.com/Office. 2019.

[11] MHTML. https://en.wikipedia.org/wiki/MHTML. 2019.

[12] Arjsoftware. http://www.arjsoftware.com/arj.htm. 2019.

[13] IQY. https://filext.com/file-extension/IQY. 2019.

[14] Working with Microsoft Publisher's PUB Format. Lifewire. https://www.lifewire.com/pub-file-in-design-software-1078142. 2018.

[15] SETTINGCONTENT-MS File Extension. Fileinfo. https://fileinfo.com/extension/settingcontent-ms. 2013.

[16] Cohen A, Nissim N, Rokach L, et al. SFEM: Structural Feature Extraction Methodology for the Detection of Malicious Office Documents Using Machine Learning Methods[J]. *Expert Systems With Applications*, 2016, 63: 324-343.

[17] Nissim N, Cohen A, Moskovitch R, et al. ALPD: Active Learning Framework for Enhancing the Detection of Malicious PDF Files[C]. *2014 IEEE Joint Intelligence and Security Informatics Conference*, 2014: 91-98.

[18] Attacking interoperability: an OLE edition. BlackHat. https://www.blackhat.com/docs/us-15/materials/us-15-Li-Attacking-Interoperability-An-OLE-Edition.pdf. 2015.

[19] Hamon V. Malicious URI Resolving in PDF Documents[J]. *Journal of Computer Virology and Hacking Techniques*, 2013, 9(2): 65-76.

[20] Nissim N, Cohen A, Glezer C, et al. Detection of Malicious PDF Files and Directions for Enhancements: A State-of-the Art Survey[J]. *Computers & Security*, 2015, 48: 246-266.

[21] Gryaznov D. Scanners of the year 2000: Heuristics[C]. *The 5th International Virus Bulletin*. 1999, 113.

[22] Schultz M G, Eskin E, Zadok F, et al. Data Mining Methods for Detection of New Malicious Executables[C]. *2001 IEEE Symposium on Security and Privacy*, 2001: 38-49.

[23] Damashek M. Gauging Similarity with N-Grams: Language- Independent Categorization of Text[J]. *Science*, 1995, 267(5199): 843-848.

[24] Stolfo S J, Wang K, Li W J. Towards Stealthy Malware Detection[M]. Malware Detection. Springer, Boston, MA, 2007: 231-249.

[25] Li W J, Stolfo S, Stavrou A, et al. A Study of Malcode-Bearing Documents[M]. Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007: 231-250.

[26] Gao Y X, Qi D Y. Analyze and Detect Malicious Code for Compound Document Binary Storage Format[C]. *2011 International Conference on Machine Learning and Cybernetics*, 2011: 593-596.

[27] Shafiq M Z, Khayam S A, Farooq M. Embedded malware detection using markov n-grams[C]. *International conference on detection of intrusions and malware, and vulnerability assessment*, 2008: 88-107.

[28] Zhang F Y, Qi D Y, Hu J L. Detection of Embedded Malware Based on C4.5 Decision Tree[J]. *Journal of South China University of Technology (Natural Science Edition)*, 2011, 39(5): 68-72. (张福勇, 齐德昱, 胡镜林. 基于 C4.5 决策树的嵌入型恶意代码检测方法[J]. *华南理工大学学报(自然科学版)*, 2011, 39(5): 68-72.)

[29] Li W, Su P R, Shi Y F. A Technique for Detecting Based on Calculation Malicious Documents of Vector Spaces[J]. *Journal of the Graduate School of the Chinese Academy of Sciences*, 2010(2):267-274.
(李伟, 苏璞睿, 时云峰. 基于空间向量计算的恶意文档检测技术[J]. *中国科学院研究生院学报*, 2010(2):267-274.)

[30] Laskov P, Šrndić N. Static Detection of Malicious JavaScript-Bearing PDF Documents[C]. *The 27th Annual Computer Security Applications Conference on - ACSAC '11*, 2011:

373-382.

[31] Maiorca D, Giacinto G, Corona I. A Pattern Recognition System for Malicious PDF Files Detection[C]. *International workshop on machine learning and data mining in pattern recognition*. Springer, Berlin, Heidelberg, 2012: 510-524.

[32] Mimicry. https://en.wikipedia.org/wiki/Mimicry. 2018.

[33] Lin J Y, Pao H K. Multi-View Malicious Document Detection[C]. *2013 Conference on Technologies and Applications of Artificial Intelligence*, 2013: 170-175.

[34] Jana S, Shmatikov V. Abusing File Processing in Malware Detectors for Fun and Profit[C]. *2012 IEEE Symposium on Security and Privacy*, 2012: 80-94.

[35] Lu X, Zhuge J W, Wang R Y, et al. De-Obfuscation and Detection of Malicious PDF Files with High Accuracy[C]. *2013 46th Hawaii International Conference on System Sciences*, 2013: 4890-4899.

[36] Chen K, Wang P, Lee Y, et al. Scalable Detection of Unknown Malware from Millions of Apps[J]. *Journal of Cyber Security*, 2016, 1(1): 24-38.
(陈恺, 王鹏, Yeonjoon Lee, 等. 面向海量软件的未知恶意代码检测方法[J]. *信息安全学报*, 2016, 1(1): 24-38.)

[37] Wang L N, Tan C, Yu R W, et al. The Malware Detection Based on Data Breach Actions[J]. *Journal of Computer Research and Development*, 2017, 54(7): 1537-1548.
(王丽娜, 谈诚, 余荣威, 等. 针对数据泄漏行为的恶意软件检测[J]. *计算机研究与发展*, 2017, 54(7): 1537-1548.)

[38] Jia X Q, Zhou G Z, Huang Q J, et al. FindEvasion: An Effective Environment-Sensitive Malware Detection System for the Cloud[C]. *International Conference on Digital Forensics and Cyber Crime*. Springer, Cham, 2017: 3-17.

[39] Smutz C, Stavrou A. Malicious PDF Detection Using Metadata and Structural Features[C]. *The 28th Annual Computer Security Applications Conference on - ACSAC '12*, 2012: 239-248.

[40] Maiorca D, Corona I, Giacinto G. Looking at the Bag is not enough to Find the Bomb: An Evasion of Structural Methods for Malicious PDF Files Detection[C]. *The 8th ACM SIGSAC symposium on Information, computer and communications security - ASIA CCS '13*, 2013: 119-130.

[41] Srndic N, Laskov P. Detection of Malicious PDF Files Based on Hierarchical Document Structure[C]. *The 20th Annual Network & Distributed System Security Symposium*, 2013: 1-16.

[42] Maiorca D, Ariu D, Corona I, et al. A Structural and Content-Based Approach for a Precise and Robust Detection of Malicious PDF Files[C]. *2015 International Conference on Information Systems Security and Privacy*, 2015: 27-36.

[43] Šrndić N, Laskov P. Hidost: A Static Machine-Learning-Based Detector of Malicious Files[J]. *EURASIP Journal on Information Security*, 2016, 2016(1): 1-20.

[44] Mao W X, Cai Z M, Tong L. Malware Detection Method Based on Active Learning[J]. *Journal of Software*, 2017, 28(2): 384-397.
(毛蔚轩, 蔡忠闽, 童力. 一种基于主动学习的恶意代码检测方法[J]. *软件学报*, 2017, 28(2): 384-397.)

[45] Nissim N, Cohen A, Elovici Y. ALDOCX: Detection of Unknown Malicious Microsoft Office Documents Using Designated Active Learning Methods Based on New Structural Feature Extraction Methodology[J]. *IEEE Transactions on Information Forensics and Security*, 2017, 12(3): 631-646.

[46] Li M, Liu Y Z, Yu M, et al. FEPDF: A Robust Feature Extractor for Malicious PDF Detection[C]. *2017 IEEE Trustcom/BigDataSE/ICESS*, 2017: 218-224.

[47] Yu M, Jiang J G, Li G, et al. Malicious Documents Detection for Business Process Management Based on Multi-Layer Abstract Model[J]. *Future Generation Computer Systems*, 2019, 99: 517-526.

[48] Yu M, Jiang J G, Li G, et al. A Unified Malicious Documents Detection Model Based on Two Layers of Abstraction[C]. *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems*, 2019: 2317-2323.

[49] Maiorca D, Corona I, Giacinto G. Looking at the Bag is not enough to Find the Bomb: An Evasion of Structural Methods for Malicious PDF Files Detection[C]. *The 8th ACM SIGSAC symposium on Information, computer and communications security - ASIA CCS '13*, 2013: 119-130.

[50] Xu W L, Qi Y J, Evans D. Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers[C]. *2016 Network and Distributed System Security Symposium*, 2016: 21-24.

[51] Toth T, Kruegel C. Accurate Buffer Overflow Detection via Abstract Pay Load Execution[C]. *International Workshop on Recent Advances in Intrusion Detection*. Springer, Berlin, Heidelberg, 2002: 274-291.

[52] Willems C, Holz T, Freiling F. Toward Automated Dynamic Malware Analysis Using CWSandbox[J]. *IEEE Security & Privacy*, 2007, 5(2): 32-39.

[53] Li W J, Stolfo S J. Thwarting Attacks in Malcode-Bearing Documents by Altering Data Sector Values[J]. *Technical Report CUCS-025-09*, Columbia University, 2008.

[54] Xu M., Kim T. PlatPal: detecting malicious documents with platform diversity[C]. *USENIX Security Symposium*, 2017: 271-287.

[55] Snow K Z, Krishnan S, Monrose F, et al. SHELLOS: Enabling Fast Detection and Forensic Analysis of Code Injection Attacks[C]. SEC'11: The 20th USENIX conference on Security. 2011: 9.

[56] Hou Yongqian. Design and Implementation of a Libemu-based Shellcode Detection System[D]. Beijing: Peking University, 2014.

(侯永干. 一个基于 Libemu 的 Shellcode 检测系统的设计与实现[D]. 北京: 北京大学, 2014.)

[57] Schreck T, Berger S, Göbel J. BISSAM: Automatic Vulnerability Identification of Office Documents[C]. *International conference on detection of intrusions and malware, and vulnerability assessment*. Springer, Berlin, Heidelberg, 2012: 204-213.

[58] Wang W P, Bai J Y, Zhang Y C, et al. Dynamic Taint Tracking in JavaScript Using Revised Code[J]. *Journal of Tsinghua University (Science and Technology)*, 2016, 56(9): 956-962,968.
(王伟平, 柏军洋, 张玉婵, 等. 基于代码改写的 JavaScript 动态污点跟踪[J]. *清华大学学报(自然科学版)*, 2016, 56(9): 956-962,968.)

[59] Ma H L, Wang W, Han Z. Detecting and De-Obfuscating Obfuscated Malicious JavaScript Code[J]. *Chinese Journal of Computers*, 2017, 40(7): 1699-1713.
(马洪亮, 王伟, 韩臻. 混淆恶意 JavaScript 代码的检测与反混淆方法研究[J]. *计算机学报*, 2017, 40(7): 1699-1713.)

[60] Wang L N, Tan C, Yu R W, et al. The Malware Detection Based on Data Breach Actions[J]. *Journal of Computer Research and Development*, 2017, 54(7): 1537-1548.
(王丽娜, 谈诚, 余荣威, 等. 针对数据泄漏行为的恶意软件检测[J]. *计算机研究与发展*, 2017, 54(7): 1537-1548.)

[61] Iwamoto K, Wasaki K. A Method for Shellcode Extractionfrom Malicious Document Files Using Entropy and Emulation[J]. *International Journal of Engineering and Technology*, 2016, 8(2): 101-106.

[62] Liu C, Xia B, Yu M, et al. PSDEM: A Feasible De-Obfuscation Method for Malicious PowerShell Detection[C]. *2018 IEEE Symposium on Computers and Communications*, 2018: 825-831.

[63] Polychronakis M, Anagnostakis K G, Markatos E P. Comprehensive Shellcode Detection Using Runtime Heuristics[C]. *The 26th Annual Computer Security Applications Conference on - ACSAC '10*, 2010: 287-296.

[64] Wu Xuefeng. Analysis of malicious PDF documents[D]. Jinan: Shandong University, 2012.
(武雪峰. 恶意 PDF 文档的分析[D]. 济南: 山东大学, 2012.)

[65] Sun Benyang. Research on Security Detection Technology of PDF Documents[D]. Shanghai: Shanghai Jiaotong University, 2015.
(孙本阳. PDF 文档的安全性检测技术研究[D]. 上海: 上海交通大学, 2015.)

[66] Lu X, Zhuge J W, Wang R Y, et al. De-Obfuscation and Detection of Malicious PDF Files with High Accuracy[C]. *2013 46th Hawaii International Conference on System Sciences*, 2013: 4890-4899.

[67] Gao X, Yu M, Jiang J G, et al. A Combined Malicious Documents Detecting Method Based on Emulators[J]. *Applied Mechanics and Materials*, 2014, 602/603/604/605: 1707-1712.

[68] Li M, Zhou Y, Yu M, et al. Combining Static and Dynamic Analysis for the Detection of Malicious JavaScript-Bearing PDF Documents[C]. *Computer Science, Technology and Application: Proceedings of the 2016 International Conference on Computer Science, Technology and Application*. 2017: 475-482.

[69] Feng D, Yu M, Wang Y. Detecting Malicious PDF Files Using Semi-Supervised Learning Method[C]. *International Conference on Advanced Computer Science Applications and Technologies*, 2017.

[70] Rieck K, Krueger T, Dewald A. Cujo: Efficient Detection and Prevention of Drive-by-Download Attacks[C]. *The 26th Annual Computer Security Applications Conference on - ACSAC '10*, 2010: 31-39.

[71] Curtsinger C, Livshits B, Zorn B G, et al. ZOZZLE: Fast and Precise In-Browser JavaScript Malware Detection[C]. *USENIX security symposium*. 2011: 33-48.

[72] Corona I, Maiorca D, Ariu D, et al. Lux0R: Detection of Malicious PDF-Embedded JavaScript Code through Discriminant Analysis of API References[C]. *The 2014 Workshop on Artificial Intelligent and Security Workshop - AISec '14*, 2014: 47-57.

[73] Liu D P, Wang H N, Stavrou A. Detecting Malicious Javascript in PDF through Document Instrumentation[C]. *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014: 100-111.

[74] Wen W P, Wang Y J, Meng Z. PDF File Vulnerability Detection[J]. *Journal of Tsinghua University (Science and Technology)*, 2017, 57(1): 33-38,43.
(文伟平, 王永剑, 孟正. PDF 文件漏洞检测[J]. *清华大学学报(自然科学版)*, 2017, 57(1): 33-38,43.)

[75] Zhang Y Q, Dong Y, Liu C Y, et al. Situation,Trends and Prospects of Deep Learning Applied to Cyberspace Security[J]. *Journal of Computer Research and Development*, 2018, 55(6): 1117-1142.
(张玉清, 董颖, 柳彩云, 等. 深度学习应用于网络空间安全的现状、趋势与展望[J]. *计算机研究与发展*, 2018, 55(6): 1117-1142.)

[76] Zhang L, Cui Y, Liu J, et al. Application of Machine Learning in Cyberspace Security Research[J]. *Chinese Journal of Computers*, 2018, 41(9): 1943-1975.
(张蕾, 崔勇, 刘静, 等. 机器学习在网络空间安全研究中的应用[J]. *计算机学报*, 2018, 41(9): 1943-1975.)

[77] Dalvi N, Domingos P, Mausam, et al. Adversarial Classification[C]. *The 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, 2004: 99-108.

[78] Biggio B, Corona I, Maiorca D, et al. Evasion Attacks Against Machine Learning at Test Time[C]. *Joint European conference on machine learning and knowledge discovery in databases*. Springer, Berlin, Heidelberg, 2013: 387-402.

[79] Xu W, Qi Y, Evans D. Automatically evading classifiers[C]. *The*

*2016 network and distributed systems symposium*. 2016, 10.

[80] Dang H, Huang Y, Chang E C. Evading Classifiers by Morphing in the Dark[C]. *The 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017: 119-133.

[81] Papernot N, McDaniel P, Goodfellow I, et al. Practical Black-Box Attacks Against Machine Learning[C]. *The 2017 ACM on Asia Conference on Computer and Communications Security*, 2017: 506-519.

[82] Alfeld S, Zhu X, Barford P. Explicit defense actions against test-set attacks[C]. *The AAAI Conference on Artificial Intelligence*. 2017, 31(1).

[83] Jana S, Shmatikov V. Abusing File Processing in Malware Detectors for Fun and Profit[C]. *2012 IEEE Symposium on Security and Privacy*, 2012: 80-94.

[84] Demontis A, Melis M, Biggio B, et al. Yes, Machine Learning can be more Secure! a Case Study on Android Malware Detection[J]. *IEEE Transactions on Dependable and Secure Computing*, 2019, 16(4): 711-724.

[85] McQueen M A, McQueen T A, Boyer W F, et al. Empirical Estimates and Observations of 0Day Vulnerabilities[C]. *2009 42nd Hawaii International Conference on System Sciences*, 2009: 1-12.

[86] Mei R, Meng Z, Huo W. Research and Implementation of Typical Document CVE Vulnerability Detection Tools[J]. *Netinfo Security*, 2014(6): 18-22.
(梅瑞, 孟正, 霍玮. 典型文档类 CVE 漏洞检测工具的研究与实现[J]. *信息网络安全*, 2014(6): 18-22.)

[87] Bai P. *Research and Implementation Of0day Vulnerability Detection Technology in Document Types*[D]. Beijing: Beijing University of Posts and Telecom, 2015.
(白鹏. 文档类型 0day 漏洞检测技术的研究与实现[D]. 北京: 北京邮电大学, 2015.)

[88] Raynal F, Delugré G, Aumaitre D. Malicious Origami in PDF[J]. *Journal in Computer Virology*, 2010, 6(4): 289-315.

[89] Albertini. This PDF is a JPEG; or This Proof of Concept is a Picture of Cats. *PoC or GTFO Ox03*. 2014.

[90] Rndic N, Laskov P. Practical Evasion of a Learning-Based Classifier: A Case Study[C]. *2014 IEEE Symposium on Security and Privacy*, 2014: 197-211.

[91] Endignoux G, Levillain O, Migeon J Y. Caradoc: A Pragmatic Approach to PDF Parsing and Validation[C]. *2016 IEEE Security and Privacy Workshops*, 2016: 126-139.

[92] Smutz C, Stavrou A. Preventing Exploits in Microsoft Office Documents through Content Randomization[C]. *International Symposium on Recent Advances in Intrusion Detection*. Springer, Cham, 2015: 225-246.

[93] Iwamoto M, Oshima S, Nakashima T. A Study of Malicious PDF Detection Technique[C]. *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems*, 2016: 197-203.

[94] PJScan. https://sourceforge.net/p/pjscan/home/Home/. 2019.

[95] Jsunpackn. https://www.aldeid.com/wiki/Jsunpackn. 2014.

[96] OfficeMalScanner. https://www.aldeid.com/wiki/OfficeMalScanner. 2013.

[97] QuickSand.io. http://www.quicksand.io/. 2017.

[98] Libemu. https://sourceforge.net/projects/libemu/. 2014.

[99] SpiderMonkey. https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey. 2019.

[100] ViperMonkey. https://github.com/decalage2/ViperMonkey. 2018.

[101] Peepdf. https://github.com/jesparza/peepdf. 2016.

[102] Origami-pdf. http://code.google.com/p/origami-pdf/wiki/GettingStarted. 2019

[103] Microsoft Office Visualization Tool. https://www.portalprogramas.com/en/microsoft-Office/visualization-tool/. 2019.

[104] OLEtools. https://pypi.org/project/oletools/. 2019.

[105] YARA. https://yara.readthedocs.io/en/v3.8.1/. 2018.

[106] Threat Analyzer. https://www.vipre.com/products/business- protection/analyzer/. 2019.

[107] PDF Examiner. http://www.pdfexaminer.com/. 2019.

[108] Vicheck. http://vicheck.ca/. 2019.

[109] VirusShare. https://virusshare.com/. 2019.

[110] VirusTotal. https://www.virustotal.com/. 2019.

[111] VirSCAN. http://www.virscan.org/. 2019.

**喻民** 现任中国科学院信息工程研究所高级工程师, 研究领域为恶意代码分析。Email: yumin@iie.ac.cn

**姜建国** 现任中国科学院信息工程研究所研究员, 研究领域为网络安全与保密技术。Email: jiangjianguo@iie.ac.cn

**Gang Li(李罡)** 现任澳大利亚迪肯大学信息技术学院副教授, 数据智能研究中心主任, 研究领域为数据安全分析。Email: gang.li@deakin.edu.au

**刘超** 现任中国科学院信息工程研究所正研级高级工程师, 研究领域为网络空间安全。Email: liuchao@iie.ac.cn

**黄伟庆** 现任中国科学院信息工程研究所正研级高级工程师, 研究领域为网络空间安全。Email: huangweiqing@iie.ac.cn

**宋楠** 现在中国科学院信息工程研究所网络空间安全专业攻读硕士学位, 研究领域为恶意文档检测。Email: songnan@iie.ac.cn