基于混合特征的恶意 PDF 文档检测

杜学绘，林杨东，孙奕
（解放军战略支援部队信息工程大学河南省信息安全重点实验室，河南 郑州 450001）

摘 要：针对现有恶意 PDF 文档在检测方案存在特征顽健性差、易被逃避检测等问题，提出了一种基于混合特征的恶意 PDF 文档检测方法，采用动静态混合分析技术从文档中提取出其常规信息、结构信息以及 API 调用信息，并基于 *K*-means 算法设计了特征提取方法，聚合出表征文档安全性的核心混合特征，从而提高了特征的顽健性。在此基础上，利用随机森林算法构建分类器并设计实验，对所提方案的检测性能以及抵抗模拟攻击的能力进行了探讨。

关键词：恶意 PDF 文档；混合特征；机器学习；检测
中图分类号：TP393
文献标识码：A
doi: 10.11959/j.issn.1000−436x.2019028

Malicious PDF document detection based on mixed feature

DU Xuehui, LIN Yangdong, SUN Yi
Henan Provincial Key Laboratory of Information Security, Information Engineering University, Zhengzhou 450001, China

Abstract: Aiming at the problem of poor robustness and easy to evade detection in the detection of malicious PDF document, a malicious PDF document detection method based on mixed features was proposed. It adopted dynamic and static analysis technology to extract the regular information, structure information and API calling information from the document, and then a feature extraction method based on *K*-means clustering algorithm was designed to filter and select the key mixed features that characterize the document security. Ultimately, it improved the robustness of features. On this basis, it used random forest algorithm to construct classifier and perform experiment to discuss the detection performance of the scheme and its ability to resist mimicry attacks.

Key words: malicious PDF document, mixed feature, machine learning, detection

1 引言

PDF 是由 Systems[1]在 1993 年提出的，用于文件交换的一种文件格式，其优点在于跨平台、能保留文件原有格式、开放标准等。PDF 文件格式自被提出以来，就以其高效性、稳定性和交互性而被广泛应用于政府、组织、企业等重要机构的日常办公。

自 2008 年第一个基于恶意代码的 PDF 漏洞[2]被提出后，针对 PDF 文件格式漏洞的利用技术迅速发展。近几年来，随着社会工程学和 APT（advanced persistent treat）等攻击手段的兴起和发展，PDF 格式的漏洞也不断被挖掘出来，利用嵌入恶意 PDF 文档的钓鱼邮件，结合社会工程学对政府、组织、企业等进行攻击的案例也屡见不鲜。

2016 年 9 月，PaloAlto 研究中心的安全研究人员首先发现了针对苹果系统的新型钓鱼邮件攻击，其技术核心在于钓鱼邮件中携带有木马病毒的恶意 PDF 文档。2017 年 5 月 12 日，WannaCry 勒索

收稿日期：2018−02−05；**修回日期**：2018−05−11
通信作者：林杨东，xd_lyd@163.com
基金项目：国家高技术研究发展计划（“863”计划）基金资助项目（No.2015AA016006）；国家自然科学基金资助项目（No.61702550）
Foundation Items: The National High Technology Research and Development Program of China (863 Program) (No.2015AA016006), The National Natural Science Foundation of China (No.61702550)

病毒软件借助高危漏洞“永恒之蓝”在全球范围内爆发，攻击范围覆盖医疗、教育、公安等多个行业，影响极其恶劣。相关人士表示，WannaCry 勒索病毒主要通过钓鱼邮件结合恶意附件进行传播，然后嵌入文档、感染文档并对文档进行加密。

种种案例表明，钓鱼邮件是当前恶意文档的主要传播途径，攻击者通常将恶意文档作为邮件附件，结合社会工程学等，诱使用户打开恶意文档，从而执行进一步的恶意操作。现有的针对恶意 PDF 文档的反病毒系统大多是基于签名的方法和基于启发式规则的方法[3]，其在应对多态攻击方面存在一定的缺陷，且无法应对新型安全威胁。为了解决这些问题，近年来，研究工作主要分为以下 2 个方面：1) 着重关注恶意 PDF 文档中的静态特征，如结构特征、内容元数据特征等，来判别恶意 PDF，而不关注其恶意内容及具体的行为操作；2) 利用静态分析和动态分析的方法，针对恶意 PDF 文档中的 JavaScript 特征进行检测。前一种方法的检测效果和检测效率均比后者更优，且可以检测不包含 JavaScript 的恶意文档，然而，这种方法由于选取特征的顽健性较弱，已被证明极易被攻击者通过简单的操作而绕过，因此，研究人员又重新关注恶意文档中 JavaScript 特征的相关研究。

为了解决现有检测方案特征顽健性较差、易被逃避检测的问题，本文提出了一种基于混合特征的恶意 PDF 文档检测方案。该方案采用静态分析技术从文档中提取出其常规信息（版本号、大小等）以及其结构信息，采用动态分析技术从文档中提取出文档执行时的 API（application programming interface）调用信息。对于常规信息，本文根据对文档的分析结果进行过滤筛选；对于结构信息及 API 调用信息，本文采用 *K*-means 聚类算法对其进行聚类，从中聚合出最能表征文档恶意性的关键结构特征及 API 调用特征，以此 3 种特征构成特征向量，利用随机森林算法构建分类器并最终设计实验进行验证。结果表明，本文方法与现有已公开的 3 种最新检测工具相比，检测率和误报率均有一定提高，且在应对基于特征加法的模拟攻击上效果较好。

2　相关工作

2.1　PDF 文档结构简介

由于 PDF 使用起来方便快捷并且支持各类功能，其在我国信息化进程中逐渐成为主流的文档信息交换的主要格式，但与此同时也成为恶意代码传播的主要载体[4]。

一个典型的 PDF 文档通常包含丰富的文本、图像等信息，此外，在 PDF 中，往往还存在超链接、数据交换以及允许其他应用调用的接口等来为 PDF 文档的各类功能提供支持。因此，PDF 拥有十分复杂的结构特征。

PDF 文档由一组逻辑上相互连接的对象组成，其结构[5]主要包括文件头（header）、文件主体（body）、交叉引用表（cross-reference table）、文件尾（trailer）和其他可选部分。文件头主要记录了该文件使用的 PDF 结构标准规范，文件主体包括了所有的对象内容，交叉引用表记录了间接对象的地址索引表，文件尾主要记录了交叉引用表的物理位置。

文件头位于 PDF 文档的最前端，指定了文件使用的 PDF 规范的版本，如“%PDF-1.7”表示该 PDF 文档使用 PDF 1.7 版本的规范。需要注意的是，文件头可以被放置在 PDF 文件的前 1 024 B 中的任意位置。

文体主体由文件头和交叉引用表之间的所有对象组成，这些对象共同构成了 PDF 文件的具体内容，如页面、文本、链接、图像等。

交叉引用表为间接对象的随机存取提供了索引地址表，它的一个重要作用是保证了在整个 PDF 文件被完整加载之前，可以将部分文档先进行显示。对于一个 PDF 文档而言，其交叉引用表会随着 PDF 文件的更新而更新，不断添加新的内容。

文件尾位于文件的结尾，以“trailer”为起始标志，指定了交叉引用表以及一些特殊对象的物理地址，并以“%%EOF”为结束标志。

2.2　恶意 PDF 文档检测相关方法

恶意 PDF 文档检测技术已有一定的基础。目前，对恶意 PDF 文档检测的分析方法主要有 2 种类型，即静态分析和动态分析。静态分析主要针对 PDF 文档中的静态特征，利用反编译的手段从文档中解析出结构、内容、JavaScript 代码等特征，并以此建立恶意 PDF 文档检测模型；动态分析则是利用虚拟机、硬件模拟等技术，通过 API 挂钩等手段对恶意 PDF 文档及其内嵌代码的动态执行进行监控，并以此建立 PDF 文档行为检测模型。

早期的恶意文档检测，普遍使用静态分析的方法。2007 年，Li 等[6]首次提出了利用机器学习结合

n-gram 分析对原始文档进行字节级分析，然而这些方法主要针对 word 文档、可执行文件（exe）等，并未在 PDF 文档检测方面进行尝试，其主要原因在于受编码、过滤和加密等技术的影响，无法定位到 PDF 文档中的主要特征。随着恶意 PDF 文档的危害性不断增强，后续的研究工作着重分析 PDF 文档中的 JavaScript 代码特征。

PDF 文档中的 JavaScript 代码特征主要有 2 类，即动态行为特征和词汇特征。针对 JavaScript 代码的动态行为特征分析已经提出了许多解决方案，并设计出了 CWSandbox[7]、JSand[8]、Cujo[9]、Zozzle[10]、Prophiler[11]等多种 JavaScript 代码分析工具，被广泛用于不同格式的文档中嵌入式 JavaScript 代码的检测，在此基础上形成了 MalOffice[12]、ShellOS[13]、MDscan[14]等检测系统。然而，利用虚拟环境对 JavaScript 进行分析的时间开销和计算开销都较高，且攻击者可以利用不同的 JavaScript 引擎或不同的阅读器版本等来绕过检测。为了降低开销，研究人员逐渐关注于对 JavaScript 代码的静态词汇分析，通过解析器，对 PDF 文档中的 JavaScript 进行定位和提取，随后利用分类器进行学习，形成检测模型，典型的代表有 PJScan[15]等。

然而，在针对 PDF 文件中 JavaScript 代码特征的分析过程中，由于存在压缩、加密、混淆等技术手段，JavaScript 代码的定位、抽取及解析始终具有较大难度，因此，针对元数据[16-17]的分析方法应运而生。基于元数据特征检测的特征抽取过程一般较为简单，且不需要对 JavaScript 进行解析和执行，其关注的特征主要有内容特征和结构特征 2 类。内容特征主要关注 PDF 文档中的重要关键字，如/JS、/JavaScrip 以及字体对象、流对象长度等一系列特征，其主要不足在于无法抽取出流对象中的数据，而流对象数据又容易被攻击者用来隐藏恶意内容，从而绕过此类检测。另一类利用 PDF 文档结构特征的检测方法[18]在检测率和应对新型威胁上均有进一步的提高，然而已被证明易被攻击者通过在正常文档中嵌入恶意内容来绕过[19]。

尽管通过对元数据的分析建立起来的检测模型的检测率及效率均较高，但其顽健性方面存在明显的不足，因而研究又逐渐转向 PDF 文档中的恶意代码。最新的进展主要有：通过在 PDF 文档中嵌入环境监控代码对 PDF 文档的动态行为进行监控[20]；根据 PDF 文档运行过程中的 API 调用信息作为特征对 PDF 文档进行分类检测[21]；利用沙箱来增强打开 PDF 文档从而保护系统的安全性等[22]。此后，Maiorca 等[23]提出了结合 PDF 文档的结构和内容特征来对恶意 PDF 文档进行检测，在一定程度上增强了纯静态的结构特征分析所带来的顽健性弱的问题，但由于采用的仍是静态分析，因此在对抗模拟攻击方面依然存在顽健性不足的问题。

3 基于混合特征的恶意 PDF 文档检测方法

通过对现有技术的分析，本文发现近几年对于 PDF 文档检测的研究大部分集中于检测其中的 JavaScript 代码以及相应的结构和内容元数据。

针对 JavaScript 代码的检测顽健性较强，但存在定位和解析 JavaScript 困难、分析代价较高、难以应对其他类型的安全威胁等问题；针对结构和内容元数据的分析检测效率和检测率较高，然而容易被攻击者设计特定文档而绕过，顽健性较差。

为了克服这些缺点，本文提出了一种基于混合特征的恶意 PDF 文档检测方法，在现有研究的基础上，着重关注恶意 PDF 文档的常规特征、结构特征及其 API 调用特征。方案框架如图 1 所示。

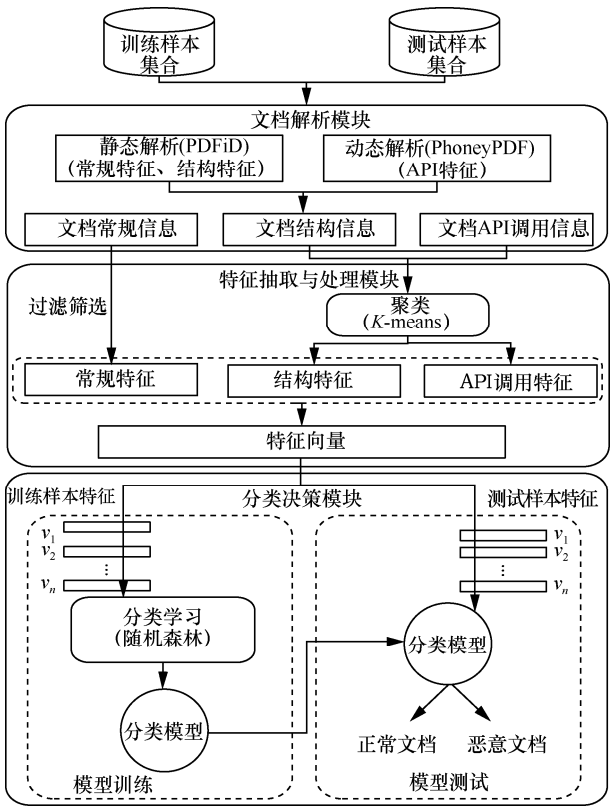


图 1 基于混合特征的恶意 PDF 文档检测方案框架

本文以 PDFiD 和 PhoneyPDF 这 2 款工具为基础构建了文档解析器，采用动静态分析相结合的方法，从文档中提取内容信息和结构信息，并将 JavaScript 特征转化为对应的 API 调用特征从而解决 JavaScript 代码定位难、代码混淆等问题，并通过聚类获得最能表征恶意 PDF 文档的结构特征及 API 调用特征，去除了大量冗余信息，并将此特征集合与常规特征相结合作为混合特征向量用于分类器的训练学习，最终得到检测模型。

3.1 混合特征设计

由于单一特征存在顽健性差等问题，因此本文设计了一种混合特征，用以表征 PDF 文档的恶意性，具体可分为常规特征、结构特征、API 调用特征 3 类，下面进行详细介绍。

3.1.1 常规特征

通过对 PDF 文档的分析，结合已有的在恶意 PDF 文档检测领域的相关经验结果，本文选取了以下 7 个重要特征作为常规特征，包括文件大小、文件版本号、包含 JavaScript 代码的对象数量、嵌入的文件数量、不完整对象的数量、交叉引用表的数量、特殊操作函数（如 OpenAction、Launch 等）数量。这些常规特征与 PDF 文档的安全相关性如表 1 所示。

这些单一特征并不足以用来标识 PDF 文档的恶意性，但它们结合起来可以作为 PDF 文档的一个整体概述。例如，根据对大量文档的分析，恶意 PDF 文档的大小通常小于正常 PDF 文档的大小，因为一般恶意 PDF 文档并不包含有用的文本、图片等内容，且文件越小，其感染效率就越高；同样，对象、交叉引用表通常用于隐藏文档中的恶意内容，与文档的恶意性存在一定的相关性；此外，JavaScript 代码及特殊操作函数等是绝大部分恶意 PDF 文档完成其恶意目的的必备内容，也能在一定程度上反映文档的恶意性。因此，这些特征构成了 PDF 文档的常规特征，其可以在一定程度上表征文档的恶意性，但并非充分条件。

3.1.2 结构特征

利用结构特征来表征 PDF 文档的恶意性，最早由 Šrndic 等[18]提出，其方案以恶意 PDF 文档在结构上与正常 PDF 文档存在的差异性为基础，设计了一种结构路径用于表征 PDF 文档的结构特征，其主要问题在于特征复杂，且不利于进一步分析。因此本文设计了一种更为简便的且更具有可解释性的结构特征。不同类别的文档，其文档结构上的关键字存在一定的差异，当一个关键字在正常样本或恶意样本中出现的频率较大时，其可在一定程度上反映文档的实际类别。因此，本文在对 PDF 文档进行解析并获取其结构关键字及其频率的基础上，设计了一种结构特征提取算法对所提取出的关键字进行聚类，从中筛选得到出现频率较高的关键字作为结构特征，从而以尽可能小的特征数量来最大程度地表征 PDF 文档。算法具体描述如下。

输入　PDF 文档样本集合 D，关键字集合 K

输出　特征关键字集合 K_t

1) 记 F 为关键字在样本中出现频次的集合

```
for each k_i in K do
    for d_j in D do
        if k_i 在 d_j 中出现 then
            f_i = f_i + 1
        end if
    end for
end for
```

2) 随机选择 2 个关键字出现频次值 f 作为初始均值向量

```
repeat
    令两簇集 C_1、C_2 为空集
    for each f_i in F do
```

表 1　常规特征与 PDF 文档的安全相关性

常规特征	安全相关性
文件大小	恶意 PDF 文档一般不包含有意义的文本图片等内容，文件大小一般较小
文件版本号	不同版本的漏洞数量及安全性存在差异
包含 JavaScript 代码的对象数量	一般情况下，恶意 PDF 文档中包含 JavaScript 的可能性远大于正常 PDF 文档
嵌入的文件数量	正常文档一般没有额外嵌入的文件，而部分类型的恶意文档通过嵌入恶意文件来实施攻击
不完整对象的数量	恶意文档中存在缺失结构字段的对象数量比例往往远大于正常文档
交叉引用表的数量	恶意文档中缺少交叉引用表或交叉引用表不可识别的比例往往远大于正常文档
特殊操作函数	恶意文档中出现 OpenAction、Launch 等操作函数的可能性较正常文档更大

计算样本 f_i 与初始均值向量的距离

根据距离将样本归入相应的簇

end for

计算新的均值向量

until 当前均值向量均无更新

3) 令 $t_1=f_t$，f_t 为簇 C_2 里靠近簇 C_1 的边界样本值

for each $f_i > t_1$ do

将对应的关键字 k_i 添加到特征关键字集合 K_t，$K_t = K_t \cup \{k_i\}$；

end for

本文利用上述算法，分别对正常 PDF 集合和恶意 PDF 集合的特征关键字集合进行提取，最终得到正常样本关键字子集合 K_b 和恶意样本关键字子集合 K_m 及其对应的出现频次，分别用以表示正常样本的结构特征和恶意样本的结构特征。本算法主要包括 3 个步骤，对应的复杂度分别为 $O(|K||D|)$、$O(2T|K|)$、$O(|K|)$，其中，T 为 K-means 算法中的迭代次数，由于迭代次数远小于样本数量，因此算法复杂度为 $O(|K||D|)$。

结构特征集合中的关键字数量主要取决于样本集合和聚类结果。文档的结构特征通常与文件执行的特征操作相关联，并且主要通过关键字来执行对应操作，因此选择特征关键字来表示文档的结构特征是可行的。例如，/Font 是正常样本中的特征关键字，主要是因为此关键字与文档中的字体相关联，出现此关键字说明样本内容中会显示不同字体，对于正常样本来说这是合理的，但恶意文档一般不包含具体内容，以轻便简单为主，因此一般不出现此关键字；又如，/OpenAction 是恶意样本中的特征关键字，其主要功能是执行对象中的打开操作，常用来执行 JavaScript 代码，这与恶意样本中有 90%以上的样本包含 JavaScript 相吻合[24]。

通过聚类，利用文档结构中的关键字特征来表征其结构特征，极大地降低了特征向量的复杂度，并且由于排除了大量冗余关键字，从而加大了攻击者在正常文档结构基础上构建恶意文档从而绕过检测的难度。

3.1.3 API 调用特征

恶意 PDF 文档中所包含的恶意代码往往都会经过复杂的混淆和隐藏，直接进行静态分析难以解决代码定位难与代码混淆等问题，为此，本文基于 API 调用与恶意代码执行过程的相关性，利用 API 调用特征来间接地表征 JavaScript 代码的执行特征。现有的针对恶意 PDF 文档中 JavaScript 代码的分析中，最常见的是采用 SpiderMonkey 等工具进行分析，这些工具最大的不足在于其识别的标准为 JavaScript 通用标准，对于 PDF 文档中特定 JavaScript 代码（如 app.doc.getAnnots、app. plugIns. length 等）无法有效识别。本文采用了 PhoneyPDF 这一分析框架对 PDF 的内嵌 JavaScript 执行过程进行分析，它是一种基于 Adobe DOM 仿真的分析框架，可以执行并分析 PDF 中所使用的各类 JavaScript 代码。通过对其执行过程中的 API 调用进行抽取，设计了一种特征 API 调用提取算法对关键 API 调用进行了选择。

记 R 为 Acrobat PDF 标准中所有可调用的 API 函数集合，N 为其元素个数。本文的任务即从全集 R 中筛选出对 PDF 文档标签具有表征价值的 API 函数子集合。

对于任意的 $r \in R$，定义 $\theta=\sum\varphi_i$ 为其有效性权值，则有

$$\varphi_i=\begin{cases}\dfrac{1}{m}, & r\text{在第}i\text{个样本中出现且样本为恶意样本}\\ -\dfrac{1}{b}, & r\text{在第}i\text{个样本中出现且样本为正常样本}\\ 0, & r\text{不在第}i\text{个样本中出现}\end{cases} \tag{1}$$

有效性权值 θ 表示对应的 API 函数在恶意样本和正常样本中出现的比率，θ 越大，说明其在恶意文档中出现的可能性越大，可作为表征恶意文档的特征；反之，θ 越小，则可将其作为正常文档的特征。

本文利用 K-means 算法，根据有效性权值 θ，计算对应的欧氏距离，对训练样本的 API 调用进行聚类，找到阈值 t_2 并将其分为 2 个簇（k=2），对应的子集 $R_t=\{r_j||\theta_j|> t_2\}$ 则为对应的特征 API 集合。算法具体描述如下。

输入 AcrobatPDF 标准中可调用 API 集合 R，恶意 PDF 集合 D_m 及其 API 调用集合 R_m，正常 PDF 集合 D_b 及其 API 调用集合 R_b

输出 API 调用特征集合 R_t

1) 计算恶意样本及正常样本数量

$M = |R_m|, B = |R_b|$

2) 计算 API 调动的有效性权值 θ

for each r_i in R do

for each d_j in D_m and d_k in D_b do

if $r_i \in R_{d_j}$ then

$\varphi = \frac{1}{M}$;

else if $r_i \in R_{d_k}$ then

$\varphi = -\frac{1}{B}$;

else then

$\varphi = 0$;

end if

$\theta_i = \theta_i + \varphi$;

end for

end for

3) 令 $\theta_i' = |\theta_i|$，随机选择 2 个样本值作为初始均值向量

repeat

令 $C_1 = \varnothing$，　$C_2 = \varnothing$;

for each r_i in R do

计算其有效性权值 θ_i' 与初始均值向量的距离

根据距离将样本划入相应的簇

end for

计算新的均值向量

until 当前均值向量均无更新

4) 令 $t_2 = \theta_t'$，θ_t' 为簇 C_2 里靠近簇 C_1 的边界样本值

for each $\theta_i' > t_2$ do

将对应的 API 调用 r_i 添加到特征调用集合 R_t，即 $R_t = R_t \cup \{r_i\}$

end for

在特征 API 调用提取算法中，*K*-means 算法的迭代轮数远小于本文的样本集合大小，因此本文算法复杂度为 $O(|R||D|)$。为了提升算法的效率，本文使用样本中出现的所有 API 调用集合来代替 AcrobatPDF 标准中可调用 API 集合 R，从而在一定程度上降低了算法的复杂度。

3.2　分类检测算法

为了对训练集中的良性和恶意样本进行学习，建立模型并利用测试集进一步优化和调整模型，本文采用半监督学习来建立检测模型。由于本文所设计的混合特征在数据结构上存在差异，因此需要选择一种能适应多种不同数据结构的分类算法。决策树算法可以较好地解决此问题，然而决策树存在容易导致过拟合、泛化效果较差等问题，因此，本文采用随机森林算法。随机森林算法是对决策树算法的一个集成和改进，它在以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入了随机属性选择，能较好地应对本文所设计混合特征存在的异构性，且计算开销小、集成的泛化性较好。最终，选择了 10 棵树的随机森林算法进行分类器的构建，并采用十折交叉验证过程。

3.3　方法评价指标

评价本文方法的功能指标主要是对恶意 PDF 文档的检测效果进行评价，恶意文档分类结果混淆矩阵定义如表 2 所示。

表 2　　恶意文档分类结果混淆矩阵

真实情况	预测结果	
	恶意文档	良性文档
恶意文档	D_{MM}'	D_{MB}'
良性文档	D_{BM}'	D_{BB}'

表 2 中，D_{MM}'表示恶意 PDF 文档被正确检测为恶意的样本数量；D_{MB}'表示恶意 PDF 文档被错误检测为良性的样本数量；D_{BM}'表示良性 PDF 文档被错误检测为恶意的样本数量，D_{BB}'表示良性 PDF 文档被正确检测为良性的样本数量。对应的几项评价指标如下。

1)真正类率（TPR, true positive rate），又称为检测率，表示被正确检测为恶意的恶意 PDF 文档数占恶意 PDF 文档总数的比率，计算式为

$$\mathrm{TPR} = \frac{D_{\mathrm{MM}}'}{D_{\mathrm{MM}}' + D_{\mathrm{MB}}'} \tag{2}$$

2)假正类率（FPR, false positive rate），又叫误报率，表示被错误检测为良性的恶意 PDF 文档数占恶意 PDF 文档总数的比率，计算式为

$$\mathrm{FPR} = \frac{D_{\mathrm{BM}}'}{D_{\mathrm{BM}}' + D_{\mathrm{BB}}'} \tag{3}$$

3)准确率（accuracy），表示检测结果是正确的样本数占样本总数的比率，计算式为

$$\mathrm{Acc} = \frac{D_{\mathrm{MM}}' + D_{\mathrm{BB}}'}{D_{\mathrm{MM}}' + D_{\mathrm{MB}}' + D_{\mathrm{BM}}' + D_{\mathrm{BB}}'} \tag{4}$$

4　实验设计与结果分析

4.1　实验样本集合与环境配置

在测试实验中，本文构建了 5 928 个恶意 PDF 文档以及 5 881 个良性 PDF 文档的实验样本集合。

其中，恶意样本主要在 VirusTotal 上收集而来，类别如表 3 所示，包括近几年来 Adobe Reader 及 Acrobat 的高危漏洞以及 2004—2011 年用户提交到 VirusTotal 的未经过鉴别分类的恶意 PDF 文档样本。这部分未鉴别的样本涵盖了常见的利用恶意 PDF 文档进行攻击的多种类型，如内嵌代码、内嵌其他恶意文件等。

表 3 恶意 PDF 文档样本类别

漏洞编号	样本数量/个
CVE-2016-4255	12
CVE-2015-5090	31
CVE-2014-0512	17
CVE-2014-0496	39
CVE-2013-0640	21
CVE-2012-0754	3
CVE-2011-2462	25
CVE-2010-2883	49
CVE-2010-0188	25
其他未归类样本	5 708

正常样本主要通过在 Google、Yahoo 上下载得到，包括论文、销售广告、报告等 PDF 文档，并通过卡巴斯基杀毒软件检测为正常样本。在此需要强调的是，本文对正常样本集合进行了控制，着重增加了包含 3D 图像、flash、视频、JavaScript 等内容的正常 PDF 样本，保证了正常样本集合的全面性，降低了由于样本不平衡性所导致的结果误差。正常 PDF 样本类别如表 4 所示。

表 4 正常 PDF 文档样本类别

类别	样本数量/个
包含 3D 图像的文档	28
包含 JavaScript 代码的文档	219
包含 flash 的文档	61
PDF 格式的 jar 文件	3
包含视频的文档	35
内嵌文件的文档	16
其他未归类样本	5 519

本文随机地将这些数据分为训练样本以及测试样本，其中，训练样本共包括 3 949 个恶意 PDF 文档和 3 916 个良性 PDF 文档；测试样本包括 1 979 个恶意样本和 1 965 个良性样本。实验的软硬件环境配置如表 5 所示。

表 5 实验环境配置

硬件配置	软件配置
2 个英特尔至强处理器（E5 2600V4）	Ubuntu
RAM（90 GB）	PyCharm

4.2 检测性能测试

为对本文所设计方案的检测性能进行验证，在训练集（3 949 个恶意 PDF 文档和 3 916 个良性 PDF 文档）的基础上，对本文的模型进行了训练，从而建立检测模型，并对测试集（1 979 个恶意样本和 1 965 个良性样本）进行测试。最终将测试结果与现有的 3 种最新公开的恶意 PDF 检测工具（PJScan、PDFRate、Wepawet）相比较。由于 PJScan 使用的是单分类 SVM 算法，其训练数据不包括正常样本，因此本文仅使用训练集以及测试集中的恶意样本作为 PJScan 的训练集和测试集，从而进行对比实验。而另外 2 款公开的工具均为在线服务，其中，PDFRate 使用了 5 000 个恶意 PDF 样本（收集于 Contagio）和 5 000 个正常样本进行训练，而 Wepawet 的训练数据集及大小均未知，因此，想要设计实验对这 2 款系统进行相同的准确训练并不可行。但通过比较这 3 类系统与本文方法对测试样本的检测结果，可以在一定程度上反映本系统的检测性能。

4 种方法对测试集最终的检测结果如表 6 所示。本文主要从检测率（真正类率）、误报率（假正类率）和准确率这 3 个方面对模型的检测性能进行比较，表 6 中括号内为标准差。

表 6 4 种方法检测效果对比

系统	检测率	误报率	准确率
本文方法	99.73% (±0.16%)	0.030%(±0.014%)	99.85%
PJScan	80.64% (±1.44%)	0.020%(±0.012%)	90.27%
PDFRate	99.05% (±0.63%)	0.092%(±0.028%)	97.39%
Wepawet	88.93% (±0.81%)	0.061%(±0.008%)	94.32%

在实验过程中发现，所有不包含 JavaScript 代码的文档均会被 PJScan 系统分类为正常文档，这是由于 PJScan 系统仅针对文档中的恶意 JavaScript 代码进行检测，但这显然是不合理的。此外，Wepawet 系统在解析结构不完整的 PDF 样本时，会出现分析出错的情况，这是由于其没有实现全部 Adobe 的规范，而仅是模拟嵌入式 JavaScript 代码和可执行文件的运行结果。

实验结果表明，本文方法在检测性能上明显优于 PJScan 和 Wepawet，PJScan 的误报率最低，但其检测率相较于本文方法而言差距较大；Wepawet 的检测率及误报率均弱于本文方法；PDFRate 在检测率方面与本文方法较为接近，而在误报率方面则存在一定的弱势。总地来说，本文方法在检测性能上要优于其他 3 类工具。

为对本文检测模型的检测效率进行进一步的分析，将本文方法的单个文件的平均检测时间与其他已公开该项数据的方案进行对比，结果如表 7 所示，其中，本文方法以及 PJScan 的数据为实验得出，其他方案的数值均为从文献[13-14,22]中获得。此外，由于 Wepawet 以及 PDFRate 为在线服务，本文无法得到其准确检测时间，并且在公开文献中也无法找到相关描述，因此本文并未对其进行分析。

表 7　　几种不同方法检测效率对比

系统	类型	检测耗时/ms
本文方法	动静态混合分析	650~1 200
PJScan	纯静态分析	58
ShellOS	纯动态分析	7 400~25 460
MDScan	动静态混合分析	1 500~3 000
结构路径	纯静态分析	28

本文方法针对正常文档的平均检测耗时为 650 ms，针对恶意文档的平均检测耗时为 1 200 ms，可以满足终端机器上的实时检测需求。在实验过程中发现，正常文档的检测耗时主要取决于文档的解析过程，这是由于正常文档往往包含丰富的内容（文本、图片、表格），而仅小部分文档包含 JavaScript 代码，其文档解析过程耗时较长；恶意文档一般只包含执行漏洞或恶意代码的对象，其具体页面内容较少，因此文档解析过程较快，其检测耗时主要取决于执行 JavaScript 的时间。

可以看到，检测效率主要与采用的分析类型有关，总体趋势为纯静态分析<动静态混合分析<纯动态分析。PJScan 和基于结构路径的方法在检测耗时上比本文方法更优，但本文方法使用了模拟 Adobe DOM，可以对 PDF 文档进行更全面的分析。此外，与同类型的 MDScan 相比，本文方法的检测耗时更少，主要原因在于本文方法间接地使用特征 API 调用来表征 JavaScript 代码特征，而 MDScan 则是对 Shellcode 进行进一步的执行与分析。

4.3　抗攻击测试

通过对本文设计的方案进行分析不难发现，攻击者要想攻击本文提出的检测系统，主要可从 3 个方面进行：1)彻底研究本文系统的检测框架，设计出本文系统无法提取到的恶意特征；2)使用恶意样本破坏本文的数据集，干扰模型的建立过程；3)分析本文选取的检测特征，进而操控样本对应的特征，最终找到可绕过检测的恶意样本。这 3 种方法均可实现对本文系统进行攻击，但很明显，前两种方法十分复杂，且需要操控本文系统的数据集，因而不易实现；而第三种方法则仅需要攻击者不断修改并提交自制恶意样本到本系统进行检测，直至被误报为正常样本，即攻击成功。

为测试本文系统对上述第三种攻击手段的防御能力，本文选择了基于特征加法的模拟攻击作为对抗模型。模拟攻击是指通过微弱地改变文件的整体结构，从而将恶意内容嵌入正常 PDF 文档中，形成相应的恶意 PDF 文档，这种攻击模型已经被证明能十分简便有效地逃避基于结构路径检测系统。为简化实验过程，本文在其思想的基础上，通过直接对已抽取出的恶意 PDF 文档特征向量进行调整，在其特征向量中加入正常 PDF 文档的相关特征，从而模拟攻击过程。此过程可以简单地理解为，通过不断增加恶意文档的正常性，来引起检测模型的误报，从而达到逃避检测的目的。

嵌入的良性特征主要从正常样本特征集合 K_b 和 R_b 中选取。K_b 和 R_b 为利用本文的特征提取算法对样本集合进行特征提取得到的集合。由于一般情况下，正常样本不包含 JavaScript 代码，嵌入对应的 API 调用特征，可能导致文档恶意性的增加，因此，本文仅选择正常样本的关键字特征进行嵌入。定义特征关键字在样本集合中出现的概率 p，平均每个样本中出现的次数为 c，则特征关键字在样本集合中出现的期望值 e 可表示为

$$e = p \times c \tag{5}$$

特征关键字的良性表征度 γ 为

$$\gamma = \frac{e_b - e_m}{e_b} \tag{6}$$

其中，e_b、e_m 分别表示特征关键字在正常样本和恶意样本中出现的期望值。

为最大限度地考量模型应对模拟攻击的能力，本文根据特征关键字的良性表征度，选取了最能表征良性 PDF 文档的 20 个特征，具体描述如表 8 所示。

表 8　　**嵌入的良性特征关键字**

特征关键字	正常样本集合		恶意样本集合		良性表征度
	出现概率	平均次数	出现概率	平均次数	
/Prev	81.63%	3.66	1.52%	3.98	0.98
/ColorSpace	63.09%	6.21	2.00%	4.63	0.98
/CropBox	81.86%	6.70	3.70%	3.53	0.98
/Linearized	77.50%	1.00	1.42%	2.45	0.96
/ProcSet	97.50%	15.50	28.00%	2.48	0.95
/PDF	97.47%	15.09	27.70%	2.48	0.95
/Metadata	73.46%	2.79	1.75%	5.87	0.95
/Font	89.37%	15.76	21.11%	3.39	0.95
/elements	71.29%	1.47	1.34%	4.34	0.94
/Resources	99.59%	15.42	31.52%	3.00	0.94
/Rotate	82.02%	6.72	19.22%	2.18	0.92
/Subtype	99.74%	26.60	75.75%	2.88	0.92
/Encoding	63.37%	6.06	16.18%	2.39	0.90
/BaseFont	65.24%	6.22	17.34%	2.37	0.90
/Length	100.00%	31.59	98.99%	4.40	0.86
/Contents	99.41%	5.49	33.59%	2.36	0.85
/FlateDecode	97.52%	22.83	82.22%	4.37	0.84
/Filter	99.90%	24.27	95.14%	4.30	0.83
/Type	100.00%	40.69	99.92%	7.75	0.81
/Parent	99.16%	11.74	96.56%	2.47	0.80

从表 8 可以看到，根据本文的选取原则，得到的特征关键字大部分与 PDF 文档中具体内容的展示、存储相关，如提纲、颜色、字体、过滤器等，这与正常 PDF 文档往往比恶意 PDF 文档包含更多的实质性内容（如文字、图片、表格等）是一致的。

本文以嵌入的特征数量为变量，在已被本文系统正确检测为恶意的 PDF 文档中，根据嵌入良性特征的数量，将恶意 PDF 文档特征向量中对应值修改为正常样本中出现的取整期望值。例如，在恶意 PDF 文档原特征向量中，上述 20 项特征的值所组成的向量为（0,0,0,0,1,1,0,1,0,1,0,0,0,0,2,0,0,1,3,0），则本文的操作为当嵌入正常样本数量为 20 时，恶意 PDF 文档特征中上述 20 项特征值将被修改为（3,4,5,1,15,15,2,14,1,15,6,27,4,4,32,5,22,24,41,12）。

通过这种方式，近似地模拟攻击者修改后的恶意 PDF 文档的特征向量，并对所构建的恶意特征向量进行检测。实验重复此过程 5 次，统计其每次攻击成功的样本数量并计算其绕过本文检测系统的平均成功率，实验结果如图 2 所示。

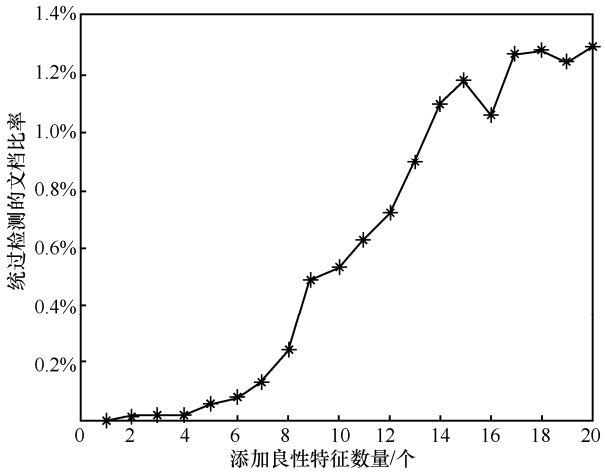


图 2　基于特征加法模拟攻击的成功率曲线

从图 2 可以看到，随着添加良性特征数量的增多，攻击成功的样本数量呈上升趋势，并最后趋于平稳，其中，成功率峰值为 1.29%，并在添加的良性特征数量超过 16 个后趋于平缓。一般地，特征向量中增加良性特征的数量越多，基于特征加法的模拟攻击成功率就会越高；所添加的良性特征越有

代表性，则其攻击成功率越高。而即便选择了 20 个最具有代表性的良性特征进行嵌入时，本文系统在应对基于特征加法的模拟攻击的检测率仍能高达 98.71%，这充分证明了所提取的混合特征的顽健性。

事实上，在现实攻击中，攻击者并不清楚本文的训练数据集以及所选取的特征，因此，其攻击过程远比上述模拟攻击复杂。攻击者需要对系统内部参数进行逆向工程，例如，通过设计梯度下降攻击来计算逃避模式，构建系统的代理副本等。为此，攻击者需要收集数据集，并且复制本文的提取特征过程，最终构建一个模拟的代理分类器。这种攻击虽然可行，但其攻击过程较复杂，且攻击代价较高。

4.4　实验局限性

实验证明，本文方法与近几年提出的 3 类公开的检测系统相比，检测率、误报率均有一定的提升，且在检测效率上能满足终端用户实时检测的需求。此外，通过设计基于特征加法的模拟攻击实验，证明了本方案所提取混合特征的顽健性。但是本文的实验仍然存在一定的不足，主要是在验证和研究良性特征数量与混合特征的顽健性、系统的检测率之间的联系缺乏实验证明。理论上，增加混合特征向量中良性特征的数量，可以有效提高检测准确率；但另一方面，良性特征的增多会导致逃避攻击的成功率增高，原因在于攻击者通过在恶意文档中添加良性特征可能会提高分类器错误分类的概率。此外，由于本文采用了聚类算法对特征进行筛选，进而构建了特征向量，特征数量取决于聚类过程，因此未设计实验对特征数量与检测准确率和顽健性之间的关系进行分析，而只是进行了定性分析，但特征数量是否最优、是否会导致过拟合，仍需要进一步的研究与讨论。

5　结束语

本文提出了一种基于混合特征的恶意 PDF 文档检测方法，通过对文档进行动静态分析，对文档基本信息、结构信息和 API 调用信息进行聚合，抽取出表征文档安全性的混合特征，并利用随机森林算法构建检测模型，在一定程度上解决了现有检测模型易被逃避检测的缺陷，提高了模型的顽健性。但本文方法还存在一些有待改进的地方：1)所选取的 3 类特征相互之间无权重关系，而实际上 API 调用特征往往更能反映文档的安全性，其权重还需通过实验进一步研究与讨论；2)抗攻击实验中未对特征数量与检测模型的准确性、顽健性之间的相互关系进行分析，无法确定模型选取的特征数量是否最优、是否会导致过拟合；3)模型的检测效果对训练数据的质量较为敏感，且存在部分无法解析的文档，需设计额外的操作进行筛选和剔除，还需进一步对解析器进行改进和优化。

参考文献：

[1] SYSTEMS A. PDF reference: adobe portable document format, version 1.3[M]. Addison-Wesley, 2000.

[2] BLONCE A, FILIOL E. Portable document format (PDF) security analysis and malware threats[J]. Images Paediatr Cardiol, 2008(2): 1-3.

[3] 陈亮, 陈性元, 孙奕, 等. 基于结构路径的恶意 PDF 文档检测[J]. 计算机科学, 2015, 42(2): 90-94.

[4] 武雪峰. 恶意 PDF 文档的分析[D]. 济南: 山东大学, 2012.

[5] Adobe Systems Incorporated. PDF reference: version 1.4 [J]. Textile Research Journal, 2003, 30: 1-10.

[6] LI W J, STOLFO S, STAVROU A, et al. A study of malcode-bearing documents[C]//International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment Springer-Verlag, 2007: 231-250.

[7] WILLEMS C, HOLZ T, FREILING F. Toward automated dynamic malware analysis using CWSandbox[J]. IEEE Security & Privacy, 2007, 5(2): 32-39.

[8] COVA M, KRUEGEL C, VIGNA G. Detection and analysis of drive-by-download attacks and malicious JavaScript code[C]//International Conference on World Wide Web. 2010: 281-290.

[9] RIECK K, KRUEGER T, DEWALD A. Cujo:efficient detection and prevention of drive-by-download attacks[C]//Twenty-Sixth Computer Security Applications Conference. 2010: 31-39.

[10] CURTSINGER C, LIVSHITS B, ZORN B, et al. ZOZZLE: fast and precise in-browser JavaScript malware detection[C]//Usenix Conference on Security. 2011: 3.

[11] CANALI D, COVA M, VIGNA G, et al. Prophiler: a fast filter for the large-scale detection of malicious Web pages categories and subject descriptors[C]//International Conference Companion on World Wide Web. 2012: 197-206.

[12] ENGELBERTH M, WILLEMS C, HOLZ T. MalOffice-detecting malicious documents with combined static and dynamic analysis[C]//Virus Bulletin International Conference. 2009: 1-37.

[13] SNOW K Z, KRISHNAN S, PROVOS N, et al. SHELLOS: enabling fast detection and forensic analysis of code injection attacks[C]//Usenix Conference on Security. 2011: 9.

[14] 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. 2011: 1-6.

[15] LASKOV P. Static detection of malicious JavaScript-bearing PDF documents[C]//Twenty-Seventh Computer Security Applications Con-

ference. 2011: 373-382.

[16] MAIORCA D, GIACINTO G, CORONA I. A pattern recognition system for malicious PDF files detection[C]//International Conference on Machine Learning and Data Mining in Pattern Recognition. 2012: 510-524.

[17] SMUTZ C, STAVROU A. Malicious PDF detection using metadata and structural features[C]//Computer Security Applications Conference. 2012: 239-248.

[18] ŠRNDIC N, LASKOV P. Detection of malicious pdf files based on hierarchical document structure[C]//The 20th Annual Network & Distributed System Security Symposium. 2013: 1-17.

[19] 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. 2013: 119-130.

[20] LIU D, WANG H, STAVROU A. Detecting malicious Javascript in PDF through document instrumentation[C]//IEEE/IFIP International Conference on Dependable Systems and Networks. 2014: 100-111.

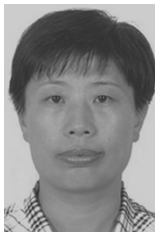
[21] CORONA I, MAIORCA D, ARIU D, et al. Lux0R: detection of malicious PDF-embedded JavaScript code through discriminant analysis of API references[C]//The Workshop on Artificial Intelligent and Security Workshop. 2014: 47-57.

[22] MAASS M, SCHERLIS W L, ALDRICH J. In-nimbo sandboxing[C]//Symposium and Bootcamp on the Science of Security. 2014: 1-13.

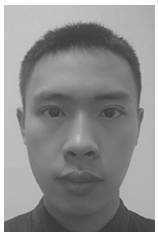
[23] MAIORCA D, ARIU D, CORONA I, et al. A structural and content-based approach for a precise and robust detection of malicious PDF files[C]//International Conference on Information Systems Security and Privacy. 2015: 27-36

[24] VATAMANU C, GAVRILUȚ D, BENCHEA R. A practical approach on clustering malicious PDF documents[J]. Journal in Computer Virology, 2012, 8(4): 151-163.

[作者简介]



杜学绘（1968− ），女，河南辉县人，解放军战略支援部队信息工程大学教授、博士生导师，主要研究方向为云计算、网络与信息安全、数据安全交换等。



林杨东（1992− ），男，广东汕头人，解放军战略支援部队信息工程大学硕士生，主要研究方向为网络与信息安全、数据安全。

孙奕（1979− ），女，河南郑州人，博士，解放军战略支援部队信息工程大学副教授，主要研究方向为网络与信息安全、数据安全交换等。