**基于分层文档结构的恶意PDF文件检测**

                                                                     Nedim Srndi c和Pavel Laskov'

认知系统系

蒂宾根大学

德国图宾根

{ nedim.srndic，pavel.laskov } @ uni-tuebingen.de

**抽象**

*恶意PDF文件在实践中仍然是大量电脑用户的真正威胁，即使在几次高调的安全事件之后。尽管Adobe和其他厂商发布了一系列安全补丁，但许多用户仍然在计算机上安装了易受攻击的客户端软件。此外，PDF格式的表现力使得攻击者可以轻松逃避检测。除了传统的防病毒产品，这些产品始终比攻击者落后一步之外，目前还没有几种方法可以用于保护最终用户系统。在本文中，我们提出了一种高性能的静态方法来检测恶意PDF文档，而不是分析JavaScript或任何其他内容，而是利用恶意和良性PDF文件的结构性质的本质差异。我们在包含约66万个真实世界恶意和良性PDF文件的数据语料库上展示其有效性，无论是在实验室条件下还是在每周重新培训的10周操作部署期间。此外，我们提出了几个学习设置关于对抗逃避抵抗的第一个比较评估，并表明我们的方法是合理抵抗复杂的攻击情况。*

**1     引言**

尽管PDF渲染软件的安全性有所提高，PDF文件仍然是“大众化”的流行攻击媒介。最近有几个大规模的攻击报告使用已知的PDF漏洞[35,15,29]。这些攻击在普通计算机用户群体中显示出相当过时的攻击媒介的惊人效力。此外，Adobe Reader应用程序中的几个新漏洞最近被发现[13]。

现有的针对基于PDF的防御机制的防御机制，

适合大规模部署的图钉仍然不足以表现力，因此也是PDF恶意软件逃避的能力。即使大多数现代防病毒产品支持PDF特定攻击的检测，检测技术仍然基于特征和严格的启发式。因此，即使后者只是对现有漏洞的微小修改，他们也不能迅速适应新的攻击媒介。

近来有关检测恶意PDF文档的研究令人吃惊，一些早期的用于检测恶意文件内容的形式化方法使用了机器学习方法和字节级的*n* - gram分析[21,32]。这些方法也适用于检测基于PDF的攻击。但是，由于它们是在2009年PDF恶意软件大流行之前发展起来的，因此无法在大型现代恶意PDF文档上进行评估。最近的大多数检测方法都基于动态分析。中这种最流行的系统为W EPAWET [[1]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn1" \o ")使用沙盒JS AND[9]，M AL ö 办公室支持 适于从CWS [11] ANDBOX[38]和S HELL OS [34]。一般来说，动态分析引入了大量的开销和延迟，因为它本质上依赖于恶意代码的执行。这激发了基于静态分析的方法[19]以及静态和动态分析的结合[37]。这两种方法都着重于检测PDF文档中的恶意JavaScript内容，这些内容涉及大量的检测逐行下载的工作; 例如[27,31,10,6]。

与之前的工作不同，我们在本文中介绍的方法旨在静态检测恶意PDF文档，*而不需要特别考虑JavaScript内容*。尽管绝大多数基于PDF的漏洞利用都依赖于JavaScript，但是存在两个主要的技术挑战，使得基于JavaScript的检测变得尤为困难。第一个挑战是在JavaScript文件中查找JavaScript内容。JavaScript可能隐藏在PDF文档的逻辑结构的深处，甚至超出了PDF标准中指定的位置[25]。PDF文档中的任何文本内容都可以使用eval（）来解释为JavaScript功能或其等价物。因此，没有什么能够阻止攻击者在文本的任何地方分发大量的JavaScript代码，并在运行时将它们组装在一起。第二个挑战在于JavaScript语言的高度表现力，这为攻击者提供了一个强大的代码混淆平台。虽然最近有人提出了一些方法来检测混淆的JavaScript [17]，但是它们的实用性还不清楚。

作为基于JavaScript的检测的替代方法，我们建议分析PDF文档的*结构属性*以区分恶意文件和良性文件。我们的方法不是寻找特定的恶意内容，而是根据其结构中恶意内容的副作用来评估文档。这种方法基于这样的直觉：由于PDF格式的复杂性，PDF文件的逻辑结构传达了大量文档的语义。因此，我们假定对结构性质的全面分析将揭示恶意和良性PDF文档之间的显着结构差异。在我们的主要贡献中，我们提出了一种新的PDF文档*结构*表示，表示为*结构路径*，可以作为各种数据分析方法自动处理的特征。

结构路径的提取可以使用现有的PDF解析器有效地执行。解析文档的结构比解释和呈现其内容*要复杂*得多。即使解析器本身可能是脆弱的，它的攻击面也比底层的渲染应用程序窄得多。由于绝大多数与PDF有关的漏洞都与JavaScript相关，所以对解析器的攻击的可能性是微不足道的，因为解析自然与JavaScript解释分离。所提出的特征提取方法可以用任何分析器进行部署。这减少了逃避技术利用文件处理中含糊的机会[16]。由于其效率，所提出的方法可以用作轻量级*内置的检测器*在相应的PDF渲染*器*内，从而完全避免了解析器的模糊性。

所提出的方法的有效性在迄今为止用于评估文件恶意软件检测器的迄今为止最大的数据语料上得到证明。我们已经从流行的恶意软件门户网站V IRUS T OTAL 收集了超过57万个恶意和良性的PDF文档，并通过一个由Google索引的90,000个良性PDF文档的实际样本来增强这个数据集。我们的实验室实验表明，该方法达到了检测率

在对恶意的V 0.0001％，假阳性率99.88％IRUS 牛逼OTAL 和良性谷歌的数据，超过在V运行任何商业杀毒IRUS 牛逼OTAL 。在良性VIRUS T OTAL 数据中观察到0.0003％的假阳性率相同的准确性，这明显偏向于可疑。在为期10周的运营部署中，每周对440,000个恶意和良性PDF文件进行再培训，所提出的方法始终优于部署在V IRUS T OTAL上的最好的反病毒引擎[[2]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn2" \o ")在7周内，由于新的数据爆发，甚至在1周内得分，在剩下的2周内输掉。这些结果证明了所提出的结构路径特征的优异检测能力，至少对于现实世界中观察到的当前PDF恶意软件是如此。

如果攻击者试图逃避我们的方法，情况会不会改变？尽管它是内容不可知的事实，所提出的方法并不是微不足道的逃避。攻击者的主要困难在于PDF格式的复杂性。即使攻击者知道哪些功能对成功检测负有责任，他也不能轻易将其删除，因为这可能会破坏攻击功能。此外，对攻击者来说，增加良性内容总是一种选择，但并不总是能够成功逃避。我们已经通过实验评估了几种潜在的*特征附加*攻击，并且观察到我们的系统中部署的一些分类算法对于这种攻击非常有效（少于0.025％的成功率）。

总之，本文提供了以下贡献：

1.    定义了一套新颖的功能，可以有效捕捉良性和恶意PDF文件之间的结构差异。

2.    使用建议的结构特点，一个classifierof PDF文档呈现，其检测精度，估计在实验室条件下约22 PDF文档，超过部署在V 43个防病毒引擎的准确性IRUS 牛逼OTAL 的时间。

3.    每周对44万份恶意和良性PDF文件进行再训练，为期10周的运行部署，证明了该方法在现实条件下的实际适用性。

4.    新分类器不依赖于对Java代码的分析，因此不会受到隐藏在PDF文档中的JavaScript新技术的影响。

5.    分类器的吞吐量被评估，并且显示出与最先进的静态检测技术相当。

6.    所提出的方法对许多规避技术的鲁棒性经实验验证。

我们的方法介绍从第二部分相关工作的回顾开始，这为本文研究的新特征和检测技术提供了动力。在第3节中总结了理解我们方法的技术方面所需的PDF格式细节。第4节介绍了系统架构和部署的算法。我们的数据语料库，旨在评估检测精度的实验结果，我们系统的吞吐量在第五部分报告。第六部分分析潜在的逃避技术并进行实验验证。最后我们在第七部分讨论我们的发现和第八部分的结论。

**2     相关工作**

以前有关检测文档恶意软件的工作与检测按路径下载的方法有许多共同的想法。这并不奇怪，因为这两种恶意软件的开发技术是相同的。现有方法可以在一定程度上重叠分类为*动态*分析方法，其中在专门仪器化的环境中打开文档，以及*静态*方法，其中执行检测而不执行恶意软件。

几个关键的思想推动了动态分析方法的发展。早期的工作遵循基于仿真的方法，其中使用抽象有效载荷执行[36]或软件仿真[1,26]来执行可疑有效载荷。但是，软件仿真没有完全覆盖指令集，因此可以被检测和回避。为了克服这个问题并提高可伸缩性，最近提出的系统S HELL OS使用硬件虚拟化而不是仿真来控制shellcode的执行[34]。S HELL OS 作为操作系统内核实现，可以有效检测应用程序分配的任何缓冲区中的shellcode。但是，这个效力是有代价的。虽然S HELLOS在检测网络级别攻击方面具有出色的带宽性能，其用于记录恶意软件的应用程序遭受高延迟（几秒钟）的影响。这种等待时间是由于检测是在应用程序分配之前必须由应用程序分配的内存缓冲器级别上进行的。

另一组动态分析方法专注于检测JavaScript执行期间的恶意行为。JS 和AND 使用了10个精心设计的启发式特征来训练良性JavaScript模型，并将攻击检测为与这些模型的大偏差[9]。类似的方法已经被成功应用于检测ActionScript 3恶意软件[24]。C UJO 建立在专门的JavaScript沙箱之上，并自动学习影响JavaScript解释器状态的事件序列模型[31]。特定于JavaScript的动态分析方法改进了集中在shellcode检测方法的性能，使其在每个文件数百毫秒的范围内，同时保持高检测精度和极低的误报率。

早期的基于*n-* gram分析的静态方法[21,32]在现代PDF恶意软件中从未被评估过。由于它们不能解决PDF格式的一些基本属性，例如编码，压缩和加密，所以可以使用类似于传统的基于签名的反病毒系统的技术，很容易地被现代PDF恶意软件所忽略。PJS CAN 是第一个证明基于异常的静态检测PDF恶意软件可行性的方法，其重点在于JavaScript内容[19]。为了提高效率，PJS CAN 的JavaScript提取器只搜索PDF标准规定JavaScript存在的位置。不幸的是，这种提取策略可以通过将JavaScript代码放入可通过PDF JavaScript API访问的任意位置，并通过类eval（）函数调用来获取。另一个最近提出的系统M ALWARE S LAYER [23]是基于模式识别方法应用于使用PDFID工具从PDF文档中提取的文本关键字。它在真实PDF数据上表现出优异的检测和误报率，但仅限于PDFID的提取功能，既不能处理多个版本号，也不能处理隐藏在对象流中的对象。PDF 率是一个最新的基于学习的静态PDF分类器，在简单的PDF元数据和字节级文件结构上运行，在具有出色分类性能的PDF文件的大型数据集上进行评估[33]。但是，它不会提取对象流，这个功能可以用来隐藏探测器的特征。

必须提到另外两个结合静态和动态分析技术的贡献。MDS CAN [37]使用PDF文件的静态分析来提取所有可用作JavaScript执行入口点的JavaScript代码块。为此，为MDS CAN 开发了一个专用解析器，它试图从文件中提取附加信息，包括从交叉引用表中省略的对象以及潜在的格式错误的对象。提取的脚本在模拟Acrobat Reader引擎的JavaScript引擎中执行。在受控执行期间，所有内存缓冲区都使用基于二进制仿真（N EMU ）的shellcode检测工具进行检查。在Z OZZLE [10]，静态和动态组件的作用是相反的。Z OZZLE 的动态部分在执行之前从Internet Explorer的JavaScript引擎中提取部分JavaScript，这自然会展现JavaScript混淆。Z OZZLE 的静态部分使用建立在检测到的JavaScript代码的语法分析之上的贝叶斯分类。相关工作的比较显示，

展示在最新的静态和动态系统中用于文档恶意软件检测。虽然动态系统具有出色的检测准确度和较低的误报率，但这些优势的代价是延迟，性能开销以及对特殊仪器环境的需求。本文提出的新方法试图从静态方面弥合这种差距，通过提高检测性能，同时保持静态方法典型的设计和计算效率的简单性。为了实现这个目标，我们开发了一个*全面*的PDF文档*静态分析*的方法，使用现成的PDF解析器（P OPPLER）。此外，我们特别关注潜在的逃避策略，并通过实验评估所提出的方法对选定攻击策略的鲁棒性。

在介绍我们系统的设计和评估之前，我们回顾一下PDF格式和文档结构的主要特征，这些特征与理解我们方法的技术方面有关。

**3     PDF文档结构**

*便携式文档格式（PDF）*是以ISO 32000-1：2008 [25]出版的开放标准，在下文中称为*PDF参考*。

PDF的语法包含四个主要元素：

•     **对象。**这些是PDF中的基本构建块。

•     **文件结构。**它指定了如何在PDF文件中布局和修改对象。

•     **文件结构。**它决定了对象在逻辑上如何组织来表示PDF文件（文本，图形等）的内容。

•     **内容流。**它们提供了一种手段来高效地存储文档内容的各个部分。

PDF中有9种基本的对象类型。简单对象类型是**布尔值**，**数字**，**字符串**和**空值**。PDF字符串的长度有限，用括号' （'和' ）' 括起来。类型**名称**用作PDF文档结构描述中的标识符。名称是使用字符“ */* ” 引入的，可以包含除*空*（0x00 ）以外的任意字符。上述5种对象类型在本文中将被称为*基本*类型。一个**数组**是用方括号“ [ '和' ] ” 括起来的PDF对象的一维有序集合。数组可能包含不同类型的PDF对象，包括嵌套数组。“ **字典”**是包含在符号“ *<<* ”和“ *>>* ” 之间的一组无用的键值对。这些键必须是*名称对象，*并且在字典中必须是唯一的。这些值可以是任何PDF对象类型，包括嵌套字典。一个**流**对象是一个PDF字典，后跟一个字节序列。字节表示可以被压缩或加密的信息，并且关联的字典包含关于是否以及如何解码字节的信息。这些字节通常包含要呈现的内容，但也可能包含一组其他对象[[3]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn3" \o ")。最后，一个**间接对象**是以前定义的对象中的任何一个，这些对象被提供了唯一的对象标识符，并被包含在关键字obj 和endobj中。由于其独特的标识符，可以通过*间接引用*从其他对象引用间接对象。

PDF对象的语法在图1的左侧示出的简化的示例性PDF文件中示出。其包含由它们的两部分对象标识符表示的四个间接对象，例如对于第一对象是1 0 ，并且obj 和endobj 关键字。这些对象是词典，因为它们被符号“ *<<* ”和“ *>>* ” 包围。第一个是*目录*字典，用它的*类型*条目表示，它包含一个带有值*目录*的PDF名称。该目录有2个额外的字典条目：*页面*和*OpenAction*。*OpenAction*是嵌套字典的一个例子。它有两个条目：*S*，表示这是JavaScript动作字典的PDF名称，以及包含要执行的实际JavaScript脚本的PDF字符串*JS*：alert（'Hello！'）; 。*Pages*是具有对象标识符3 0 的对象的间接引用：紧跟在Catalog之后的Pages字典。它有一个整数*Count*，表示文档中有2个页面，并有一个数组*Kids*用方括号标识，带有两个对Page对象的引用。使用相同的对象类型来构建剩余的页面对象。请注意，每个Page对象都包含对*Parent*项中的Pages对象的反向引用。总共有三个引用指向同一个间接对象，3 0 ，Pages对象。

各种基本对象之间的关系构成了PDF文件的逻辑树状*文档结构*，如图1的中间部分所示。文档结构中的节点本身就是对象，边对应于子对象的名称驻留在父对象中。对于数组，父子关系是无名的，并且对应于单个元素的整数索引。请注意，严格来说，文档结构不是树，而是一个有向的，可能循环的图，因为间接引用可能指向文档结构中任何位置的其他对象。文档结构中的根节点是一个特殊的PDF字典，包含名称*Catalog*的强制性*Type*条目。任何基本类型的对象构成了文档结构中的一个叶子。

以下列表显示了来自真实良性PDF文件的示例性结构路径：

|  |
| --- |
| E:\bdfirst\2018\2018Q1\Detection of Malicious PDF Files Based on Hierarchical Document Structure.files\image002.gif  **图1. PDF结构的各种表示形式：物理布局（左），逻辑结构（mid-** |

**dle）和一组结构路径（右）**

/元

/类型

/页/童装

/ OpenAction /目录

/ StructTreeRoot / RoleMap

/页/童装/内容/长度

/ OpenAction / d /资源/ ProcSet

/ OpenAction / d

/页/计数

/页面布局 ...

在第5.3.5节中提到的实验中了解到，这些是文件中存在的结构路径最能表明文件是良性的，或者缺少表示文件是恶意的。例如，恶意文件不太可能包含元数据以最小化文件大小，打开文档时不会跳转到文档中的页面，而且格式不正确，因此缺少路径，例如/ Type 和/ Pages /计数。

以下是在同一个实验中学习到的来自真实世界恶意PDF文件的结构路径列表：

/ AcroForm / XFA

/姓名/ JavaScript的

/姓名/ EmbeddedFiles

/姓名/ JavaScript的/名称

/页/童装/类型

/ StructTreeRoot

/ OpenAction /类型

/ OpenAction / S

/ OpenAction / JS

/ OpenAction ...

我们发现，恶意文件在打开文档时倾向于执行存储在多个不同位置的JavaScript，并且可以使用Adobe XML Forms Architecture（XFA）表单，因为恶意代码也可以从那里启动。

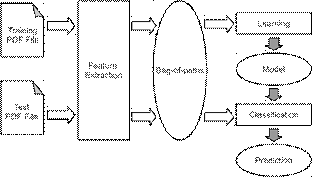
在下一节中，我们将介绍分析文档结构所需的一般方法和技术手段，以便可靠地区分恶意和不良PDF文档。

**4     系统设计**

所提出的用于基于结构的检测恶意PDF文档的方法包括以下两个步骤，如图2所示：

1.     *提取结构特征。*作为基本的预处理步骤，PDF文档的内容被解析并转换成特殊的形式，*包 - 路径*，以一种定义明确的方式表征文档结构。

2.     *学习和分类。*检测过程是由恶意和良性的PDF doc-



**图2.系统架构**

uments。在*学习*阶段，根据已知标签的数据（“训练数据”）创建一个模型。该模型编码恶意和良性数据之间的差异。在*分类*步骤中，将模型应用于新数据（“测试数据”），将其分类为恶意或良性。

下面介绍这两项基本任务的技术实现。

**4.1    特征定义**

设计数据驱动的安全工具的一种常用方法是手动定义一组随后用于学习和分类的“固有特征”。它被成功应用于网络入侵检测[20,22]，僵尸网络检测[12]，检测下载驱动[9,10,6]等相关问题。定义用于检测恶意PDF文档的功能的挑战在于PDF格式的复杂结构。因此，我们偏离了上述的知识驱动策略，并考虑更丰富的捕捉PDF复杂性的潜在特征。这些功能稍后将根据可用数据自动缩减为更小的子集。

PDF文档结构分析的最终目标是恢复其对象之间的所有亲子关系。PDF文档的树状结构可以用从根到叶的一组路径来表示，如图1的最右边部分所示。在形式上，我们定义一个*结构路径*作为沿着边缘遇到的名字的连接导致一个特定的叶子。为了符号方便，我们将使用正斜杠符号“/”作为结构路径上名称之间的分隔符[[4]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn4" \o ")。如果相同的路径跨越某些数组并导致不同的叶对象，则文档中可能会出现多次相同的结构路径。经验证据表明，文档中特定路径的计数构成了衡量不同文档之间结构相似性的一个很好的度量。这激发了选择一组结构路径作为我们系统的内在特征。

由于PDF文档中间接引用的广泛使用，多个结构路径可能会导致相同的对象。间接引用甚至可能形成循环依赖，在这种情况下，结构路径的集合变成无限的。在一些PDF的语义结构（例如页面树）中，为了便于内容呈现，*需要*多个路径。间接引用的精确处理只能通过有向图来实现。由于图的比较计算困难，所以我们坚持文档结构的树状视图，并在下面的部分引入附加的启发式方法，这些方法产生一组有限的结构路径，同时保持现有关系的合理的语义近似。

因此，要在我们的特征提取步骤中执行的主要操作是计算文档中的结构路径。额外的转换，被称为“嵌入”，可以应用于路径计数。二进制嵌入检测到非零计数的存在，频率嵌入将计数划分到文档中的路径总数，计数嵌入指的是路径计数本身。所有三种嵌入都进行了实验评估，*二进制*嵌入是由于其稍好的检测性能而选择的。

**4.2   PDF文档结构的提取**

4.1节中定义的结构特征的提取必须符合以下要求：

**R1：**必须提取所有路径的确切数量。

**R2：**该算法必须具有可重复性和鲁棒性，即必须为具有相同逻辑结构的PDF文件生成相同的一组路径。

**R3：**给定对象的多条路径之间的选择在语义上应该是关于PDF参考[25]最有意义的选择。

作为提取过程的第一步，使用PDF解析器P OPPLER[[5]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn5" \o ")解析文档。P OPPLER 的主要优势在于对PDF中各种编码的可靠处理以及从压缩流中可靠地提取对象。原则上，任何其他强大的PDF解析器都适合于提取结构路径，而我们选择P OPPLER 只是因为它的可用性和易安装性。解析器维护文档的内部表示并提供对单个对象的所有字段的访问。从概念上讲，路径提取相当于从*目录*开始的文档结构中叶子的递归枚举由解析器返回的对象。将提取的路径插入合适的数据结构（例如哈希表或地图）中以累积结构路径的计数。

必须对这个通用算法进行几个改进，以确保它终止并满足上述要求。

要求**R1**自然地被我们的特征提取的递归性质所满足。由于我们的递归只有在遇到叶子节点时才终止，所以算法保证永远不会低估特定路径的数量。为了防止由于多个路径而高估路径计数以及由于循环引用引起的无限递归，必须强制执行要求**R3**。

需求**R2**和**R3**的执行是紧密耦合的，最终依赖间接引用的智能处理。很明显，不能总是取消引用，因为这可能导致无限递归。因为算法几乎不能超越根节点，所以也不能避免它们的去参照。因此，必须实施一个一致的选择性去参照策略。

在我们的提取算法中，我们通过在叶子对象的枚举中维护广度优先搜索（BFS）顺序来处理这些问题。这个策略假设给定叶的最短路径在语义上是最有意义的。例如，这个观察直观地适用于各种情况，如图1中的例子所示，当循环关系通过字典中的*Parent*条目从明确的向上引用中产生时。虽然我们没有进一步的证据支持这个观察，我们的经验BFS遍历总是产生有意义的路径。

另外两个技术细节对于实施BFS遍历至关重要。在遍历和回溯期间，每当一个对象被访问多次时，跟踪所有访问的对象至少一次是很重要的。还需要按照某种固定的顺序对字典中的所有条目进行排序，然后再下降到节点的子节点。由于“PDF参考”不需要特定的字典字段顺序，因此必须人为地执行此类顺序以满足要求**R2**。

**4.3   学习和分类**

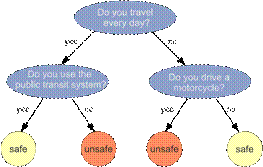
一旦提取了结构路径集上的计数或其他嵌入，几乎所有的学习算法都可以应用于从给定的训练中创建模型

数据并使用这个模型来分类未知的例子。对于合适的算法的概述，读者可以参考机器学习的任何标准教科书，例如[4,14]，或使用任何入门级的机器学习工具箱，如SHOGUN [[6]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn6" \o ")或WEKA [[7]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn7" \o ")。提供一个全面的实验证据来证明哪种机器学习方法最适合用于检测使用结构路径的恶意PDF文档，这超出了本文的范围。我们选择了两个具体的算法，决策树和支持向量机，由于主观原因在下一节介绍了各自的方法和高层次的描述。

尽管所选择的两种方法原则上都适用于高维数据，但是我们决定为了计算的原因人为地降低其维数，只选择在我们语料库中的至少1000个文件中出现的那些顺序路径（参见第5.1节在我们的实验评估中使用的数据的详细描述）。这将我们实验室实验中的特征（即结构路径）数量从900万个减少到6087个。我们没有使用类信息来选择“判别性特征”，例如在Z OZZLE中[10]。这种手动预选功能为特定的数据集引入了一种人为的偏见，并为攻击者提供了一个简单的机会，通过将相反类别的特征添加到其恶意示例中来逃避分类器。

**4.3.1****决策树**

决策树是一种流行的分类技术，其中预测是在一系列单属性测试中进行的。每个测试要么给一个例子指定一个类，要么调用进一步的测试。决策树来自操作决策领域，对于安全应用程序尤其具有吸引力，因为它们为特定的决策提供了明确的理由 - 这是安全管理员所赞赏的一个特征。图3显示了一个决策树的例子，它将一个人是否可能参与交通事故。



**图3.一个决策树的例子**

*自动决策树推理*的目标是从标记的训练数据构建决策树。决策树推理存在几种经典算法，如CART [5]，RIPPER [7]，C4.5 [28]。我们选择了一个现代的决策树推理实现C5.0，为实际应用提供了许多有用的特性，如自动交叉验证和类加权[[8]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn8" \o ")。它也可以将决策树转换成便于大型决策树视觉检查的规则集。

**4.3.2      支持向量机**

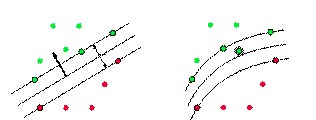
支持向量机（SVM）是另一种流行的机器学习算法[8]。它的主要几何思想，如图4所示，是将一个超平面拟合到数据中，这样两个类的例子就以最大可能的边界*M*被分开。在线性决策函数的情况下，它由超平面的权重向量*w*和阈值*ρ来表示*，它们直接用于将标签*y*分配给未知示例*x*：

*Ý*（*X*）= *瓦特*⊤*X* - *ρ*

非线性决策函数也可以通过对输入数据应用非线性变换，将其映射到具有特殊属性的特征空间（即所谓的*再生核希尔伯特空间（RKHS））中*。支持向量机的优雅在于，通过选择合适的比较两个例子*x* 1和*x* 2的非线性*核函数k*（*x* 1 *，x* 2 ），可以隐含地完成这种转换。对于对*偶* SVM学习问题的解决方案*α*，相当于原始解决方案*w*可以用于非线性判定函数，该函数表示为训练数据中的未知示例*x*与所选示例*x i*的比较，所谓的“支持向量”（在图4中用黑圈标记）：

*Ý*（*X*）= X *α 我 Ý 我 ķ*（*X，X 我*） - *ρ*

∈∈ *SV*



**图4.线性和非线性SVM**

SVM学习的有效实现可以在各种机器学习包中获得，包括WEKA和SHOGUN。在我们的实验中，我们使用了一个众所周知的独立SVM实现LibSVM [[9]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn9" \o ")。

**5     实验评估**

本节中介绍的实验的目标是测量我们提出的方法的有效性和吞吐量，以及在实际条件下的操作适用性。此外，我们将我们的方法的分类性能与其他现有的检测恶意PDF文件的方法进行比较：PJS CAN 和已建立的反病毒工具。我们的评估基于大约66万个真实世界的PDF文档的广泛数据集。

**5.1    实验数据集**

数据集的质量对于有意义的模型的推断以及任何数据驱动方法的引人注目的评估都是至关重要的。对于我们的评估，我们共获得了658,763个良性和恶意的PDF文件（大约595 GB）。我们的数据集是从Google和VIRUS T OTAL [[10]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn10" \o ")收集的，这是一种在线服务，用于检查普通用户使用大多数可用的防病毒程序上传的病毒文件。收集的数据包括以下6个数据集：

**D1：V IRUS Ť OTAL 恶意**，一种含有选自V得到的38207（1.4 GB）恶意PDF文件IRUS Ť OTAL 2012年3月5日和22日，这是由至少5个反病毒标记为恶意之间的18天期间，

**D2：VIRUS T OTAL 恶意新**，           含11,409

（527 MB）从VIRUS T OTAL 获得的恶意PDF文件2个月后，在33天内，2012年6月23日至6月24日期间，至少有5个防病毒软件标记为恶意，

**D3：VIRUS T OTAL 良性**，在2012年3月5日至22日期间，在18天的过程中从V IRUS T OTAL 获得了79,200（75GB ）的良性PDF文件，其被所有的抗病毒标记为良性，

**D4：Google良性**，包含从2007年2月5日至2012年7月25日期间2000天内1000次谷歌搜索PDF文件（没有搜索关键字）获得的90,384（73 GB）良性PDF文件

**D5：恶意操作**，从2012年7月16日至10月21日期间，从V IRUS T OTAL 获得32,526（2.7 GB）的恶意PDF文件，至少有5个防病毒软件标记为恶意软件，

**D6：良性操作**，在14周内（2012年7月16日至10月21日）从VIRUS T OTAL 获得407,037（443 GB）的良性PDF文件，所有的抗病毒标记为良性。

在V IRUS 牛逼OTAL 数据包括来自世界各地，这使我们尽可能接近真实世界的私人PDF数据尽可能人们所使用的PDF文件。事实上，良性VIRUS T OTAL 数据甚至有偏向于恶意，因为用户通常上传他们觉得可疑的文件。通过Google搜索获得的数据集消除了对良性数据恶意的偏见，并尝试捕获在该数据库上找到的平均良性PDF文件的特征

互联网。

请注意，我们认为只有当VIRUS T OTAL 文件被至少5个防病毒标记为恶意文件时才是恶意文件。被标记为恶意的1到4个AV的文件完全从实验中被丢弃，因为如经验验证的那样，对它们的正确标记几乎没有信心。由于所有PDF文件缺乏可靠的基本事实，我们假设防病毒产品的误报率为零，并且在假阳性方面不能与所提出的方法进行公平比较。

**5.2    实验方案**

设计了两种类型的实验来评估所提出的方法的检测性能：*实验室*和*操作*实验。

三个实验室实验运行静态数据，在特定的时间点捕获，训练和分类数据混合使用*5倍交叉验证*[[11]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn11" \o ")：

•    该**标准**实验的目的是对已知的恶意和平均良性数据评估了该方法的整体效益。为此，我们使用V IRUS T OTAL 恶意数据集（**D1**）和Google良性数据集（**D4**）。

•    该**可疑**实验的目的是对普通用户不信任PDF文件评估了该方法的有效性。对于这个实验，我们使用V IRUS T OTAL 恶意数据（**D1**）和V IRUS T OTAL 良性数据（**D3**）。由于良性数据偏向于恶意，本实验中的分类任务比**标准**实验更难。

•    该**WithJS**实验的目的是使我们的方法来PJS比较CAN 。对于这个实验，使用了用于**标准**实验（**D1**和**D4**）的数据集的一个子集，仅包含那些包含PJS CAN 可以提取的直接嵌入JavaScript的文件; 即30,157个恶意文件和906个良性文件。

相反，在两个操作实验中，对训练时完全不存在的文件（即，稍后获得的文件）执行分类：

•    在**小说**上训练的实验时，我们的评估对新的恶意威胁方法*过时的*训练集。对于这个实验，我们将在**标准**实验中学习到的模型应用到两个月前的新的VIRUS T OTAL 恶意数据（**D2**）中。新的良性数据没有被评估，因为它观察到的变化在这个时间跨度并不显着。

•    该**10Weeks**实验的目的是评估我们的方法的分类性能在真实世界，一天到一天的实际设定操作，并把它比作最好V的结果IRUS 牛逼OTAL 在同一时间段杀毒。这个实验是对来自操作良性（**D6**）和恶意（**D5**）的数据执行的）数据集，包含在14周的过程中收集的文件。实验每周进行一次，从第5周开始进行10周。在每一次运行中，都会对过去4周收集的文件进行特征选择。基于这些特征和数据从头开始学习新模型; 这个模型用来分类本周获得的文件。因此，在第1周到第4周期间获得的数据被用于学习将第5周收集的数据分类的模型，第6周使用第2周至第5周等。

请注意，在实践中，随着新的标记数据变得可用，定期对分类模型进行重新训练并没有根本性的困难。部署在最终用户系统上的模型可以以类似于传统防病毒系统中签名更新的方式进行更新。如5.4节所示，SVM足够有效，可以从头开始对模型进行定期的再培训。然而，我们的决策树学习算法实现缺少所需的计算性能，并且在本实验中未被评估。

**5.3    实验结果**

决策树学习算法和SVM都在我们的实验室实验中进行了评估。对于SVM，我们选择了*γ* = 0 的径向基函数（RBF）核*。*0025 和*成本*参数 *C*= 12 ，基于经验预评估。

**5.3.1      标准实验**

表1显示了**标准**实验中两种分类算法的检测结果。上面部分通过汇总所有五个交叉验证运行的结果来显示混淆矩阵（具有正确和错误分类的正面和负面文件的数量）。底部显示等性能指标：在*真*和*假阳性率*和整体*检测精度*。

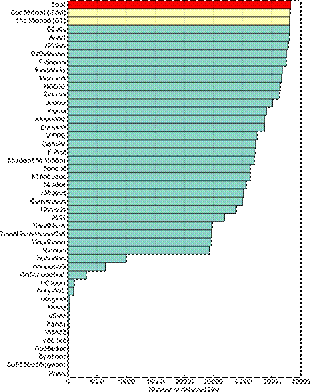
|  |  |  |
| --- | --- | --- |
|  | **决策树** | **SVM** |
| 真正的积极 | 38102 | 38163 |
| 误报 | 51 | 10 |
| 真正的消极 | 90783 | 90824 |
| 虚假否定 | 105 | 44 |
| 真正的利率 | 0.99725 | 0.99885 |
| 假阳性率 | 0.00056 | .00011 |
| 检测精度 | 0.99879 | 0.99958 |

**表1. 标准实验的汇总结果**

该**标准**实验评估我们的方法在实验室条件下的整体性能。如表1所示，尽管SVM略优于决策树学习算法，但两种算法都表现出优异的分类性能。检测准确率达到99.8％以上，假阳性率低于0.06％。

这个实验提出了我们的方法如何与现代的完全最新的商业防病毒产品进行比较的问题。在数据收集时，我们能够获得在VIRUS T OTAL 提供的43个AV引擎的检测结果。需要注意的是，由于V IRUS T OTAL 使用其命令行界面来运行AV，所以AV的检测能力仅限于静态文件处理。

图5显示了使用决策树和SVM 的V IRUS T OTAL AVs和我们的方法的真正比率。由于上述限制，我们的两种算法在这方面都优于商业防病毒引擎，多达30个防病毒引擎至少漏掉了15％的威胁。



**图5.我们的方法与商业抗病毒药物的比较**

**5.3.2      可疑的实验**

表2显示了**可疑**实验的结果。与**标准**实验相比，两种算法的分类性能确实降低了，但是性能下降非常小。

|  |  |  |
| --- | --- | --- |
|  | **决策树** | **SVM** |
| 真正的积极 | 38118 | 38163 |
| 误报 | 68 | 27 |
| 真正的消极 | 79132 | 79173 |
| 虚假否定 | 89 | 44 |
| 真正的利率 | 0.99767 | 0.99885 |
| 假阳性率 | 0.00086 | 0.00034 |
| 检测精度 | 0.99866 | 0.99939 |

**表2. 可疑实验的汇总结果**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **决策树** | **SVM** | **PJScan** |
| 真正的积极 | 30,130 | 30149 | 21695 |
| 误报 | 14 | 12 | 1 |
| 真正的消极 | 892 | 894 | 905 |
| 虚假否定 | 27 | 8 | 8462 |
| 真正的利率 | 0.9991 | 0.9997 | 0.7194 |
| 假阳性率 | 0.0154 | 0.0132 | .0011 |
| 检测精度 | 0.9986 | 0.9993 | 0.7275 |

**表3. WithJS 实验的汇总结果**

**5.3.3      WithJS实验**

表3显示了我们的算法和PJS CAN的详细实验结果。由于PJS CAN 执行异常检测; 即只使用一个类的例子（恶意）学习，PJS CAN 的实验协议略有不同。与我们的方法相同的数据子集上采用了五倍交叉验证，不同之处在于良性训练文件未被用于PJS CAN 。

总体而言，由于数据集中的强烈的类不平衡性，我们的方法比**标准**实验的表现要差一些。但是，它明显优于PJS CAN ，这是一种专门用于检测恶意JavaScript的方法，即使在为其精心挑选的数据集上也是如此。PJS CAN 在这个实验中的高假阴性率可以归因于它没有处理复杂的混淆。PJS CAN 报道的有效性与之前VIRUS T OTAL 数据[19] 的结果相当。

上述实验结果与文献报道的PDF恶意软件检测结果相比有了显着的改善，总结见下页的表4。可以清楚地看出，所提出的方法（称为S TR P ATH ）的检测性能明显好于先前报道的结果。其最接近的竞争对手M ALWARE S LAYER [23]获得了类似的真实阳性率，但其代价是假阳性率增加了200多倍。动态分析方法MDS CAN [37]和S HELL OS [34]都不会产生误报[[12]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn12" \o ")但只能检测到80-90％的恶意软件; 还应该注意的是，这些结果是在数量级较小的数据集上进行测量的。

|  |  |  |
| --- | --- | --- |
|  | **决策树** | **SVM** |
| 真正的积极 | 10681 | 10870 |
| 虚假否定 | 728 | 539 |
| 真正的利率 | 0.9361 | 0.9527 |

**表5. 新实验的汇总结果**

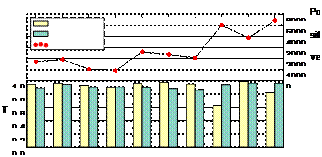
**5.3.4     新型实验**

表5显示了**Novel**实验的结果[[13]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn13" \o ")。

该**新型**实验表明，即使当我们的学习算法在2到3个月大的数据集的训练，他们依然达到可观的93％（决策树）或95％（SVM）真阳性率。

**5.3.5     10Weeks实验**

图6显示了我们的方法与**10Weeks**实验中最好的VIRUS T OTAL 抗病毒[[14]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn14" \o ")在真阳性率（TPR）方面的比较。最好的AV达到了*92.81％*的整体TPR ，明显好于我们的方法达到的*87.14％*。然而，我们的方法在10周内持续超过最佳的AV，而在第8周有一个平局。我们的方法在第14周，最值得注意的是12的性能下降促成了进一步的调查，发现了一个奇怪的趋势在V IRUS T OTAL 提交。



5 6 7 8 9 10 11 12 13 14

周

**图6.我们的方法与10Weeks 实验中最好的抗病毒的真实阳性率的比较**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | S TR P ATH | MDS CAN | PJS CAN | 小号HELL OS | M ALWARE S 层 | | 恶意样本数量 | 38207 | 197 | 30157 | 405 | 11157 | | 良性样本数量 | 90,384 | 2000 | 906 | 179 | 9989 | | 真正的正面率 | 0.9988 | 0.8934 | 0.7194 | 0.8024 | 0.9955 | | 假阳性率 | 0.0001 | 0 | .0011 | N / A | 0.0251 |   **表4.所提出的方法与以前的结果的比较** |

图6的上半部分显示了由至少5个AV检测到的VIRUS T OTAL 的PDF提交的数量，即每周的正数。在2012年10月的第一周，第12周，VIRUS T OTAL看到积极投稿数​​增加约。150％，从2000左右到5000左右。这种提交的提交水平一直持续到实验结束。仔细观察本周提交的文件，发现有两组文件，其中一组是1842，另一组是2,595，它们在字节上有所不同，但具有相同的PDF结构，即组中的每个文件对应于相同的路径。此外，两组文件的结构之间有很高的相似性。小组的小路由99条结构路径组成，所有这些路径都存在于另一组中。二者只有在更大的群体中存在额外的11个结构路径才有所不同。文件路径相同的文件也在稍后提交，但不在本周之前提交。*模糊技术*。

第12周假阴性率为37％的原因是我们的方法将小组中的1842个文件全部错误地标记为良性。所有文件的预测都是相同的，因为它们都转换成相同的行包路径，即相同的数据点。在这种情况下的一个数据点的错误分类导致多于一个假阴性率1 */* 3 ，因为测试数据是由一个源产生非常相似的，模糊化的PDF严重倾斜。在大约20个案例中，这些模糊的PDF也被所有的AV所忽略。

与较小文件组的路径对应的数据点位于SVM决策边界的错误一侧，尽管非常接近它。另外11条结构路径将决策边界上对应于较大组的路径的数据点显着地定位到正类中。其原因在于11个增加的结构路径中有8个是学习到的SVM模型中强有力的恶意指标[[15]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftn15" \o ")。在第12周后的几个星期内，这些例子在学习阶段出现，被正确分类。

第14周的性能下降来自我们的解析器无法打开的提交的文件数量（超过900）。这个异常没有被进一步调查，因为这些格式不正确的PDF对PDF渲染器应用程序没有危险，但仍然被无知的AV或解析器错误扫描，在这种情况下，更新或修复解析器，甚至使用完全不同的，因为该方法是解析器不可知的。

我们的方法在这个实验中的整体假阳性率是0.0655％，如在实验室测试中。根据我们的实验的定义，AV没有误报，因为“未定”文件（1到4次检测之间的文件）被过滤。

**5.4    吞吐量**

在处理大量PDF数据时，高吞吐量是一个重要的考虑因素，就像V IRUS T OTAL ，大公司或政府那样。我们的系统旨在处理这种负载，并利用现代计算机系统的并行处理能力。我们测量了使用决策树和SVM 对数据集**D1**， **D2**，**D3**和**D4**进行特征提取，学习和分类所花费的时间。测量是在具有8 GB的RAM的四核CPU以及先前清除了内存缓存的7,200 RPM SATA硬盘上进行的。

特征提取是最耗时的操作，因为它需要从硬盘加载所有的PDF文件。它使用7个并行处理来执行。对上述数据集中的150 GB数据进行特征提取总共花费了121分55秒，其中对恶意文件花了5分13秒，良性文件花了116分42秒。这产生了168 Mbit / s的吞吐量。

用于学习和分类的号码因决策树和SVM而不同。他们在表6中列出。

由于5次交叉验证运行中的每一个都训练了80％的训练数据，我们将所有运行的执行时间总和除以4，以获得单个训练对整个数据集需要多长时间的估计。分类时间是5个个人分类的简单总和，

|  |  |  |
| --- | --- | --- |
|  | **学习** | **分类** |
| 决策树 | 6分31秒 | 52S |
| SVM | 1分23秒 | 54S |

**表6. 标准实验的学习和分类所需的时间**

因为每个处理20％的测试数据。请注意，执行交叉验证并行运行会随着进程数量线性增加性能。尽管决策树比SVM慢得多，但总体运行时间是由特征提取决定的。

在**标准**实验中使用支持向量机的特征提取，学习和分类所需的总时间为74.4GB，数据集**D1**和**D4**为1小时和2秒，总吞吐量约为169Mbit / s，平均处理时间为每个文件28毫秒。高性能的数字是通过静态检测和并行执行来实现的。相比之下，动态的方法，如MDS CAN （每个恶意文件略少于3000毫秒，平均每个良性文件1500毫秒）和S HELL操作系统（平均每个文件5.46秒进行分析，再加上缓冲区提取的2（良性）到20（恶意，非ROP）秒）需要更多的时间。唯一的其他完全静态的方法，PJS CAN ，每个文件需要23毫秒，因为它只提取了一个定义完整的，有限的整个PDF文件的子集。

**6     逃避**

每个提出的安全措施的一个重要方面就是逃避的难度。鉴于对安全问题的学习方法的应用越来越感兴趣，以前的一些工作已经解决了学习算法的安全分析方法[3,18,2]。遵循这样的方法，在下面的部分，我们提出并实验性地评估一种新型的攻击方式，这是由攻击者的结构特征所强加的特定约束所激发的。

PDF格式的灵活性以及Acrobat Reader对规则的强制执行给了攻击者充分的机会来影响文件的内容和结构。攻击者的唯一基本约束是需要传递恶意内容并通过适当的攻击媒介触发其执行。在我们的逃避模型中，我们假设攻击者已经制作了一个恶意的PDF文件，该文件被正确地分类为恶意文件。攻击者的目标是修改文件，使其被分类为良性。我们假设攻击者*不能通过删除部分恶意内容来减少检测足迹*因此仅限于添加分类模型认为良性的内容。虽然我们无法验证这个限制对于攻击者来说是无法克服的，但直观地说，向PDF文档添加任意良性内容要比改变现有文档的语法结构容易得多，使得它仍然被观察者“正确”渲染，触发利用。

在我们的分析中，我们假设攻击者对检测机制及其分类模型有完整的认识。虽然后面的假设可能看起来过于严格，但是没有什么能阻止攻击者收集类似于训练用的替代数据集，训练他自己的分类器，从而获得对分类模型的良好猜测。或者，如果我们的方法将被部署在最终用户系统上，那么包含学习模型的文件将被分发给最终用户，就像防病毒定义一样。这将通过逆向工程使攻击者完全可以使用它们。在本节的其余部分中，我们分析了在我们的系统中部署的不同分类算法对功能增加攻击的潜在影响。

为了避开决策树分类器，攻击者必须修改他的恶意文件，这样会改变决策路径并导致错误的否定。为了实现这个目标，攻击者检查决策树，并找到确切的决策，导致终端节点的正确分类为恶意。然后，攻击者必须沿着这条路径回溯，并且如果测试是肯定的，则找到直接通向良性类别的终端节点的第一个非终结节点。如果文件中存在某个特征（如二进制嵌入的情况）或者特征的计数大于特定阈值（计数嵌入），则测试为肯定的。在这种情况下，攻击者通过向PDF文件添加适当的内容来将该特征“添加”到特征向量中通常很简单。这迫使决策树分类器做出错误的决定。如果该功能无法轻易添加到该文件（严格执行PDF结构的某些部分），则攻击者可以继续搜索以获得更好的功能选择。如果在直接导致良性终端节点而不是直接导致子树的路径上没有正面的决定，攻击者仍然可以修改他的文件，插入这样的特征，得到一个新的决策路径并重复搜索过程。这个算法可以很容易地自动化。我们能够通过在真实世界的恶意PDF样本中增加一个先前不存在的特性来验证其有效性，这迫使我们的决策树对其进行错误分类。使用随机森林而不是单一的决策树会使计算更难以逃避检测，

可以为线性SVM设计不同的攻击策略。回想一下，其决策函数计算为*ý*（*X*）= *瓦特*⊤*X* - *ρ*。参数*w*和*ρ*由模型给出，攻击者控制输入向量*x*。攻击者的目标是将预测从正向改为负向。为计数嵌入，这可以容易地通过拾取的尺寸实现*Ĵ*具有最负重量*瓦特Ĵ*在模型和插入*X*＆OverBar; *Ĵ*对应于特征的内容组块*Ĵ*使得*y*（*x*）变成负数。简单代数揭示了*X*＆OverBar; *Ĵ* =（*Ý*（*X*） - *ρ*）*/ w的Ĵ*。我们已经在实践中成功地验证了这个躲避攻击。如果最不利的特征不能被插入或者只能被插入一次，攻击者可以找到具有第二最重负的特征，依此类推。对于二进制嵌入，攻击者必须插入多个特征，但这只是一个小实际的限制。

使用RBF核退出SVM比线性情况下存在根本上更难的问题。原因在于径向基函数是非线性的，这使得修改输入向量产生负面结果的任务成为非凸优化问题。我们成功用于逃避线性支持向量机的例子并不适用于RBF。作为一个更有效的策略，我们决定试图用*模仿攻击*来逃避，通过伪装恶意文件看起来良性。在我们的情况下，这可以通过将一些良性文件的所有特征复制成恶意文件来完成。为了验证这一策略在实践中的有效性，我们随机抽取了5000个恶意和5000个良性训练文件，训练了一个RBF SVM，然后对另一个随机选择的5,000个恶意和5,000个良性测试文件进行了分类。与分离超平面距离最大的良性文件; 即最自信地被分类为良性的人被选为伪装，以使得成功攻击的机会最大化。我们将该文件的所有功能都复制到测试集中的所有5,000个恶意特征向量中。结果，假阴性的数量从28个增加到了30个; 即 最强的可能模仿攻击在5000个例子（0.025％）中仅添加了2个错误分类！对于同样设置的线性支持向量机，模仿攻击能够将检测精度降低到50％左右。这个实验实际上证明了RBF支持向量机对模仿攻击的强大抵抗力。

**7     讨论**

以上提出的潜在回避技术的实验评估和分析明确表明PDF文档的结构对于检测恶意文档具有很强的判别能力。考虑到结构性差异只是恶意内容的副作用，这一发现令人惊讶。尽管原则上可以预料，攻击者可能会找到一种在恶意结构中隐藏恶意内容的方法，但是我们对几种逃避策略的实验表明，这个任务可能比我们想象的更难。我们可以设想的最积极的逃避策略是成功的，只有0.025％的恶意实例通过使用二进制嵌入的带有RBF内核的现成的非线性SVM分类器进行测试。目前，我们没有一个严格的数学解释这样一个惊人的鲁棒性。我们的直觉表明，攻击者的主要困难在于，在他控制下的输入特征，即PDF文档的结构元素，只与分类器所使用的真实特征松散相关。真实特征的空间“隐藏”在数学上难以逆转的复杂非线性变换之后。这个观察结果证实了这样一个事实，即对二进制嵌入的线性分类器进行相同的攻击具有50％的成功率; 因此，RBF分类器的鲁棒性必须根植于其非线性变换中。我们的直觉表明，攻击者的主要困难在于他所控制的输入特征，即PDF文档的结构元素，与分类器使用的真实特征只有松散的关系。真实特征的空间“隐藏”在数学上难以逆转的复杂非线性变换之后。这个观察结果证实了这样的事实，即用二元嵌入对线性分类器进行的相同的攻击具有50％的成功率; 因此，RBF分类器的鲁棒性必须根植于其非线性变换中。我们的直觉表明，攻击者的主要困难在于，在他控制下的输入特征，即PDF文档的结构元素，只与分类器所使用的真实特征松散相关。真实特征的空间“隐藏”在数学上难以逆转的复杂非线性变换之后。这个观察结果证实了这样的事实，即用二元嵌入对线性分类器进行的相同的攻击具有50％的成功率; 因此，RBF分类器的鲁棒性必须根植于其非线性变换中。这个观察结果证实了这样一个事实，即对二进制嵌入的线性分类器进行相同的攻击具有50％的成功率; 因此，RBF分类器的鲁棒性必须根植于其非线性变换中。这个观察结果证实了这样的事实，即用二元嵌入对线性分类器进行的相同的攻击具有50％的成功率; 因此，RBF分类器的鲁棒性必须根植于其非线性变换中。

学习模型的解释与逃避攻击的成功之间的比较使我们得出这样的假设，即*对于逃避的可解性和稳健性是学习方法的对抗性特征*。事实上，最自然的解释方法，即决策树，是最容易逃避的方法。事实上，决策树的逃避是如此的微不足道，以至于我们不推荐将它们用于任何安全相关的应用程序。线性SVM的模型可以很好地解释（例如，参见[30,31]，用于安全应用中线性SVM的示例性解释）。另一方面，具有最好鲁棒性的RBF核的SVM几乎不可能解释。需要进一步的研究工作，以便更好地了解具体学习方法的这种权衡和发展的实际妥协。

**8     结论**

我们已经提出了基于良性和恶意PDF文件的基本结构性质之间的差异来检测恶意PDF文件的新方法。通过依赖结构而不是实际的内容，它使得它不必处理非常富有表现力的PDF混淆技术，JavaScript的解释和动态执行 - 相关方法继续挣扎的难题 - 并且获得了保持静态方法的好处：非常高的吞吐量和鲁棒性。

实验评估已经证明了我们的方法在实验室和操作实验中对大约66万个良性和恶意PDF文件的非常大的数据集的优良行为。它与文献（PJS CAN ，MDS CAN ，S HELL OS和M ALWARE S LAYER）最近的相关工作）。所提出的方法已经被证明对于新型攻击非常有效，即使对于真正的PDF恶意软件，在分类模型创建两个多月之后才能保持高检测精度。在实际情况下每周进行为期10周的运营部署，每周进行一次再培训，表明了所提方法的出色表现，同时也显示了及时应对突发变化的难度。在未来的工作中，将会研究提高强非平稳数据性能的技术。

所提出的方法的计算效率与先前已知的最快的静态检测方法（PJS CAN ）相当，并且达到169Mbit / s的吞吐量。这种性能比混合静态/动态方法高出一个数量级，比已建立的动态检测方法高出近两个数量级。由于所提出方法的高效率，从V IRUS T OTAL 和谷歌搜索收集的150 GB数据集可以在两个小时内处理。

在我们对潜在规避策略的分析中，我们特别关注了特征添加攻击的情况，即攻击者去除恶意内容的能力是有限的，但可以自由添加良性内容以避免被发现。我们的分析和实验评估表明，在我们的框架中部署的分类器，如决策树和线性支持向量机，可以很容易地被特征加入击败，而其他的，如带有RBF核的支持向量机，几乎免疫复杂的攻击场景。

本文提出的研究结果揭示了几个重要的公开问题。令人惊讶的事实是，PDF文档的结构为恶意内容的存在提供了强有力的指示，要求深入研究PDF格式的语法，以了解其特定功能与已知攻击的语义的相关性。此类调查还应该针对基于结构的检测方法进一步探讨潜在的逃避策略。虽然我们的实验显示，一些非线性分类算法对于特征附加攻击可以是强壮的，但详细考虑PDF语义对于理解内容修改或特征去除攻击是必不可少的。

**参考**

[1]       P. Akritidis，E. Markatos，M. Polychronakis和K. Anagnostakis。STRIDE：通过指令序列分析进行多态滑板检测。在*信息安全第20届国际会议*，375-392页，2005年。

[2]       M. Barreno，B. Nelson，A. Joseph和J. Tygar。机器学习的安全性。*Machine Learning*，81（2）：121-148，2010。

[3]       M. Barreno，B. Nelson，R. Sears，A. Joseph和J. Tygar。机器学习可以安全吗？在*ACM信息，计算机和通信安全研讨会上*，第16-25页，2006。

[4]       CM Bishop。*模式识别和机器学习*。施普林格，2007年。

[5]       L. Breiman，J. Friedman，J. Olshen和C. Stone。*分类和回归树*。Wadsworth，1984。

[6]       D. Canali，M. Cova，G. Vigna和C. Kruegel。Prophiler：用于大规模检测恶意网页的快速过滤器。在*万维网（WWW）的国际会议上*，第197-206页，2011年。

[7]       W.科​​恩。快速有效的规则归纳 在*国际机器学习会议（ICML）*，第115-123页，1995年。

[8]       C. Cortes和V. Vapnik。支持矢量网络。*Machine Learning*，20：273-297,1995。

[9]       M. Cova，C. Kruegel和G. Vigna。检测并分析驱动器下载攻击和恶意JavaScript代码。在*万维网（WWW）的国际会议上*，第281-290页，2010。

[10]    C.Curtsinger，B。Livshits，B。Zorn和C. Seifert。ZOZZLE：快速而准确的浏览器内JavaScript恶意软件检测。在*USENIX安全研讨会*，33-48页，2011年。

[11]    M. Engelberth，C. Willems和HT MalOffice - 分析各种应用程序数据文件。在2009 年的*Virus Bulletin国际会议上*。

[12]    G. Gu，P. Porras，V. Yegneswaran，M. Fong和W. Lee。BotHunter：通过IDSdriven对话关联检测恶意软件感染。在*USENIX安全研讨会*，第167-182,2007页。

[13]    谷歌警告使用的Adobe Reader -特别是在Linux上。http://www.h-online.com/open/news/item/Googlewarns-of-using-Adobe-Reader-particularly-on-Linux1668153.html。

[14]    T.黑斯蒂，R. Tibshirani，和J.弗里德曼。*统计学习的要素：数据挖掘，推理和预测*。史普林格系列统计。Springer，New York，NY，2009.第2版。

[15]    Vorsicht       bei      angeblicher telekom -onlinerechnung。http://heise.de/-1545909。

[16]    S. Jana和V. Shmatikov。滥用恶意软件检测器中的文件处理以获得乐趣和利润。在*IEEE安全与隐私专题讨论会上*，第80-94页，2012年。

[17]    S. Kaplan，B. Livshits，B. Zorn，C. Siefert和C. Cursinger。“nofus：自动检测”+ string.fromcharcode（32）+“obfuscated”.tolowercase（）+“javascript代码”。技术报告，微软研究院，2011。

[18]    P.Laskov和M.Kloft。机器学习的定量安全分析框架。在*第二次ACM研讨会上，AISec*，第1-4页，2009年11月。

[19]    P. Laskov和N. Srndi c。“静态检测恶意的带有JavaScript的PDF文档。在*年度计算机安全应用会议（ACSAC）*，第*373-382*页，2011年。

[20]    W. Lee，S. Stolfo和K. Mok。建立入侵检测模型的数据挖掘框架。在*IEEE安全与隐私研讨会上*，第120-132页，1999。

[21]    W.-J. Li，S. Stolfo，A. Stavrou，E. Androulaki和A.Keromytis。一个关于携带文档的文档的研究。在*侵入和恶意软件和脆弱性评估（DIMVA）的检测*，第231-250,2007。

[22]    M. Mahoney和P. Chan。学习异常检测敌对网络流量的规则。在*数据挖掘国际会议（ICDM）*，2003年。

[23]    D.Maiorca，G。Giacinto和I. Corona。用于恶意pdf文件检测的模式识别系统。2012年第510-524页。

[24]    TV Overveldt，C. Kruegel和G. Vigna。FlashDetect：ActionScript 3恶意软件检测。在*入侵检测（RAID）最近Adances*，274-293页，2012。

[25]    PDF参考。

http://www.adobe.com/devnet/pdf/pdf reference.html，2008。E:\bdfirst\2018\2018Q1\Detection of Malicious PDF Files Based on Hierarchical Document Structure.files\image011.gif

[26]    M. Polychronakis，K. Anagnostakis和E. Markatos。使用运行时启发式技术进行全面的shellcode检测。在*年度计算机安全应用会议（ACSAC）*，第287-296页，2010年。

[27]    N. Provos，P. Mavrommatis，M. Abu Rajab和F. Monrose。你所有的iFRAME都指向我们。在*USENIX安全研讨会*，第1-16页，2008年。

[28]    J. Quinlan。*C4.5：机器学习程序*。Morgan Kaufmann，1992。

[29]    黑洞犯罪软件套件驱动卷筒纸威胁尖峰。http://www.theregister.co.uk/2012/01/26/ sophos fakeav conficker /。E:\bdfirst\2018\2018Q1\Detection of Malicious PDF Files Based on Hierarchical Document Structure.files\image011.gif

[30]    K. Rieck，T. Holz，K. Willems，P. Dussel和P.Laskov.¨学习和分类恶意软件行为。In *Deteus of Intrusions and Malware＆Vulnerability Assessment（DIMVA），第5届国际会议*，第*108-125*页，2008年7月。

[31]    K.里克，T.克鲁格，和A. Dewald。Cujo：高效的检测和预防驱动下载攻击。在*年度计算机安全应用会议（ACSAC）*，第31-39页，2010年。

[32]    Z。Shafiq，S. Khayam和M. Farooq。使用markov n-grams嵌入式恶意软件检测。在*入侵和恶意软件和脆弱性评估的检测（DIMVA）* 2008年，88-107页。

[33]    C. Smutz和A. Stavrou。使用元数据和结构特征进行恶意PDF检测。在*年度计算机安全应用会议（ACSAC）*，2012年。要出现。

[34]    KZ Snow，S. Krishnan，F. Monrose和N. Provos。ShellOS：启用代码注入攻击的快速检测和取证分析。在*USENIX安全研讨会上*，2011。

[35]    PDF   恶意软件     作者        不断  瞄准     漏洞。http://www.symantec.com/connect/blogs/pdf-malwarewriters-keep-targeting-vulnerability。

[36]    T.托特和C. Kruegel。准确的缓冲区溢出检测通过抽象有效载荷执行 在*入侵检测（RAID）最近Adances*，274-291页，2002年。

[37]    Z.Tzermias，G.Sykiotakis，M.Polychronakis和E.Markatos。结合静态和动态分析来检测恶意文件。在*欧洲研讨会系统安全（EuroSec）*，2011。

[38]    C.威廉姆斯，霍尔兹和F.弗莱林。CWSandbox：实现自动动态二进制分析。*IEEE安全与隐私*，5（2）：32-39,2007。

[[1]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref1" \o ") http://wepawet.iseclab.org/

[[2]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref2" \o ")应该注意的是，在VIRUSTOTAL上只部署了反病毒引擎的命令行版本。

[[3]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref3" \o ")此功能原本是为了以压缩形式存储小对象的集合。但是，它已经成为攻击者混淆文档结构的流行工具。

[[4]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref4" \o ")从技术上来说，*null*是PDF名称中不允许的唯一字符，因此是结构路径中唯一合适的分隔符。

[[5]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref5" \o ") http://poppler.freedesktop.org/，v.0.14.3。

[[6]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref6" \o ") http://www.shogun-toolbox.org/

[[7]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref7" \o ") http://www.cs.waikato.ac.nz/ml/weka/

[[8]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref8" \o ") http://www.rulequest.com/see5-info.html，v.2.07。

[[9]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref9" \o ") http://www.csie.ntu.edu.tw/~cjlin/libsvm/，v.3.12。

[[10]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref10" \o ") https://www.virustotal.com/。

[[11]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref11" \o ") 5-倍交叉验证的工作原理如下：我们随机我们的数据分成5个分离子集，恶意各自含有第五和良性文件五分之一之一。学习和分类重复五次，每次选择四个子集的不同组合用于学习，剩下的一个用于分类。这个实验协议使我们能够*对每个文件进行一次分类，*同时确保*没有在分类阶段中处理的文件在相应模型的学习阶段被使用*。

[[12]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref12" \o ")奇怪的是，在假阳性率没有数据被报道小号HELLOS虽然良性数据进行收集。我们认为它是零，因为没有理由相信在一些现实的良性数据上会报告注入shellcode的错误检测。

[[13]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref13" \o ")请注意，这个实验缺少一些信息，如真正的负数，因为它只适用于恶意数据，因为良性表现的变化可以忽略不计。

[[14]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref14" \o ")最好的AV的名字没有被公开，因为在这个实验中，在TPR的5％的边界内有几个AV，他们的排名可能会逐周改变。

[[15]](file:///E:\\bdfirst\\2018\\2018Q1\\Detection%20of%20Malicious%20PDF%20Files%20Based%20on%20Hierarchical%20Document%20Structure.htm" \l "_ftnref15" \o ")线性SVM被训练用于表现出对于较小的组文件的相同的错误分类问题此功能解释的目的。通过对所有特征的权重进行计算和排序，评估过程类似于第6节中给出的线性SVM的规避策略。