EURASIP期刊

DOI 10.1186 / s13635-016-0045-0

Hidost：一种基于静态机器学习的恶意文件检测器

NedimŠrndic'1 \*和Pavel Laskov 2

抽象

从个人计算的早期开始，恶意软件即恶意软件一直是信息安全领域的持续威胁。最近的有针对性的攻击广泛使用不可执行的恶意软件作为隐身攻击媒介。在检测不可执行的恶意软件方面存在大量以前的工作，包括静态，动态和组合方法。虽然静态方法执行速度更快，但它们的适用性迄今仅限于特定的文件格式。

本文介绍Hidost，这是第一个基于静态机器学习的恶意软件检测系统，旨在以*多种文件格式*进行操作。扩展了以前发布的高效方法，它将文件的逻辑结构与其内容相结合，以实现更好的检测准确性。我们的系统已通过PDF和SWF（Flash）两种格式实施和评估。由于其模块化设计和通用功能集，可扩展到逻辑结构按层次结构组织的其他格式。在包含440,000 PDF和40,000 SWF文件的时间戳数据集的实际实验中评估，Hidost胜过了网站VirusTotal部署的所有防病毒引擎，以检测最高数量的恶意PDF文件并在SWF恶意软件中名列前茅。

**关键字：**机器学习，安全，恶意软件检测，文件格式，PDF，SWF

**1      介绍**

闯入计算机系统最有效的工具之一仍然是恶意软件，即恶意软件。自从个人计算曙光以来，它一直是一个众所周知的瘟疫，但恶意软件在近十年来已经发展出几种阴险的特性来满足犯罪业务的需求。其中之一是用于在企业和个人之间交换文件的众所周知格式的文件感染。这种感染为攻击者提供了以下好处：

1.   引诱用户打开文档比启动可执行程序更容易。

2.   文件查看者近年来一直在观察到新的漏洞，因为它们的高复杂性又造成了文件格式的复杂性。

3.   文件格式的灵活性和多功能性，使混淆嵌入的恶意内容成为可能。

相同的功能也阻碍了恶意文件的识别并增加了检测工具的计算负担。

|  |
| --- |
|  |
|  |  |

攻击者使用的最喜欢的格式是PDF（针对Adobe Reader），Flash（针对Adobe Flash Player）和Microsoft Office文件[1，2]。在2012年，开创性的漏洞攻击工具Blackhole专门针对Java，PDF和Flash文件，其继任者继续这种做法[3]。2013年，通过网络提供的不可执行恶意软件主要是针对Adobe Reader和Microsoft Office应用程序的PDF和Flash文件[2]。近期Flash已经广泛部署恶意广告，即通过广告网络在合法网站上放置恶意软件。甚至一些最着名的网站也成为这种攻击的受害者[3]。尽管普遍用于重定向到提供漏洞工具包的网站，但Flash文件直接针对Flash Player并不罕见。

非可执行文件作为有*针对性的攻击*手段尤其受欢迎。近年来，针对政府和工业界的一系列高调针对性攻击，并且越来越普遍且越来越隐蔽。针对欧洲政府机构的Miniduke针对性攻击活动使用了利用Adobe Reader零日漏洞的复杂PDF文件。微软Office中的四个不同的零日漏洞被用于针对国防工业的Elderwood攻击。APT1或CommentCrew小组针对政府和行业目标在Adobe Reader和Microsoft Office中使用了0天漏洞[4]。在2014年发现的24天0天内，有16个针对Adobe Reader和Flash Player（参见图1），而Microsoft Word文件主导了用于目标攻击的文件类型列表[1,5]。2015年前9个月，

检测恶意不可执行文件的主要困难是理解复杂格式的必要性。虽然这种困难在基于动态分析的方法中被边缘化，即在仪表化的沙箱中渲染文件，但这些方法通常相当缓慢。静态分析方法以其高性能而闻名，通常会部署临时特定于格式的检测技术，这些检测技术并未在各种格式中进行推广。为了缓解这个问题，我们提出了一种*新的静态分析方法*，它有可能在各种格式中更具可移植性。我们的实验证明，通过合并适当的格式解析器，它可以应用于PDF和Flash文件。在介绍该方法的主要特点之前，我们回顾一下相关的工作。

**1.1      相关工作**

早期关于PDF恶意软件检测的工作主要集中在磁盘上PDF文件的n-gram分析[7,8]。但是，PDF是一种复杂的文件格式[9]。PDF文件，特别是恶意文件，通常使用*混淆*以压缩的形式，使用不同的编码，甚至加密。因此，只有完整的PDF解析器才能正确解除混淆。在这方面，n-gram方法过于简单。第一种使用PDF解析器的方法是PJScan [10]。它基于嵌入在PDF文件中的JavaScript代码的词法属性使用异常检测。但是，它无法处理在运行时加载的JavaScript代码或不首先使用JavaScript的恶意软件。随后提出了两种简单的基于学习的方法，Malware Slayer [11]和PDFrate [12]，均使用基于PDF文件原始字节的启发式特征。目前提出的所有方法通常称为*静态方法*因为它们不会执行或模拟PDF文件的任何部分。根据他们对PDF文件的解析是否符合PDF标准[9]（深）或不（浅），它们可以分为深层和浅层方法。PJScan是迄今为止唯一提出的深层方法。浅层方法的一个常见弱点是PDF格式物理结构伪造的相对容易性，以PDFrate [13]为例。*所有*纯静态方法的一个共同缺点是它们无法检测动态加载的威胁，例如，当分析的文件不包含攻击代码，而是通过网络或其他文件加载它时。

随着所描述的静态方法，*动态*开发了一些方法来利用通过观察运行时打开PDF文件的影响而获得的附加信息。不依赖检查PDF文件，动态方法不受PDF混淆和物理结构伪造的影响。早期的方法基于软件仿真[14,15]。然而，软件仿真显示出易于逃避和计算密集型。其他流行的动态方法包括Wepawet [16]基于沙箱JSand [17]和基于CWSandbox [18]的MalOffice。Snow等人 提出采用硬件虚拟化并评估他们的系统ShellOS [19]的PDF恶意软件。Tang等人 在低级硬件功能上使用异常检测[20]。虽然动态方法往往比静态方法更精确，它们的执行时间使其不足以实时检测繁忙网络上的恶意文档。此外，构建和维护动态检测器能够模拟每个版本的易受攻击的软件产品，并结合每个支持的操作系统和库的每个版本，这是一项成本高昂且技术上具有挑战性的任务。另一方面，从检测器中省略目标软件的一个组合就足够了，并且为该特定版本设计的威胁将不会被检测到。

为了达到静态方法的速度*和动态*方法的精度，随后开发了*静态和动态组合*方法。MDScan执行静态JavaScript提取和动态代码执行[21]，但使用未记录功能模拟PDF JavaScript API的复杂性阻止了完整且无错误的解决方案。另一方面，MPScan可以挂载到Adobe Reader中以获得完美的JavaScript提取和反混淆处理，但是会执行静态漏洞检测[22]。由于其设计，它仅适用于单个版本的Adobe Reader上的恶意软件检测，其动态组件需要几秒钟才能运行。

与迄今为止提出的全自动方法相比，Nissim等人 建议使用*主动学习*方法，其中人类专家手动标记机器学习算法的有趣样本，目标是使检测器保持最新的威胁[23]。他们概述了一种结合了签名检测和目前描述的多种方法的设计，但将其实施和评估留给未来的工作。有关许多提及的PDF恶意软件检测方法的更详细的调查，我们请读者参考[23]。

与PDF相比，近年来仅有两种方法提供了有关检测Flash恶意软件的研究。OdoSwiff系统从2009年开始使用基于启发式的方法对静态和动态分析获得的特征[24]。它于2012年由FlashDetect成功实现，它将检测从ActionScript 2升级到ActionScript 3漏洞，并用朴素贝叶斯分类器取代了基于阈值的方法[25]。这两种方法都基于经验方法，努力编码领域专家（即恶意软件分析师）关于SWF开发现有方式的知识。这些*专家功能*表现非常好。例如，FlashDetect的机器学习分类器使用仅包含每个类别的47个样本的训练数据集进行评估，但即使这个小样本大小也足以实现高检测精度。然而，正如作者所指出的那样，一些基于启发式的特征对于承诺的逃避者来说并不健全。此外，嵌入式恶意软件可能会根据其与Adobe Flash Player的差异来检测所采用的动态执行环境，并将其行为视为一种反应。相反，这里提出的方法使用*数据驱动的方法*而不是专家功能，其检测基于良性和恶意SWF文件之间的结构差异。通过保持与漏洞无关的特性，它仍然对新型攻击持开放态度，其静态方法可以加快执行速度，并且不会受到运行时间的影响。

**1.2       贡献**

所提出的检测方法基于对分级文档结构的分析，并且此后缩写为Hidost。它是Šrndic和Laskov在[26]中发表的以前着作的延伸，在此称为SL2013。SL2013中引入的新颖性是*逻辑结构的运用*用于表征恶意和良性PDF文件。PDF逻辑结构是由PDF标准定义的高级构造，它将基本的PDF构建块组织为功能性文档。在[26]中发表的结果表明，恶意文件的属性可以从其逻辑结构中准确确定，例如JavaScript的存在和最少使用良性内容。作为一种深度静态方法，SL2013受到困扰浅层方法的PDF混淆和物理结构伪造的影响较小。根据包含66万PDF文件的真实世界数据集进行评估，SL2013展示了检测性能和吞吐量的组合，在防病毒引擎和已发布的科研工作中保持无与伦比。尽管如此，在时间戳数据的实际滑动窗口实验中，显示SL2013的检测性能不一致。它的特征定义创造了一个可被开发者利用的盲点，其超大的特征集为支持更多内存密集型机器学习分类器带来了困难。

Hidost继承了SL2013的所有优点。它在PDF文件中保持了近乎完美的检测性能和高吞吐量，该文件专门针对繁忙网络上的集中部署SL2013进行了量身定制。作为深度静态方法的进一步优势，Hidost不受PDF混淆和物理结构伪造的影响。

此外，Hidost解决了我们稍后发现的SL2013的某些缺点。特别是，我们开发了*结构路径整合*（SPC），这是一种用于合并相似特征的技术。这样的合并特征可以更好地保留逻辑结构的语义并减少特征集对特定数据集的依赖性。SPC的好处有三个：（a）逃避的攻击面减少; （b）随时间变化的特征集是有限的; 和（c）功能的数量急剧减少。总之，这些改进使得Hidost比SL2013更安全和实用。

然而，最重要的是，本文介绍了一种新颖的Hidost系统设计，使其能够推广到多种不相关的文件格式。据我们所知，Hidost是第一个适用于多种文件格式的基于静态机器学习的恶意软件检测系统。它的通用性是通过将基于PDF逻辑结构的特征定义扩展为具有层次逻辑结构（Flash的SWF格式）的第二种文件格式实现的。最后，更进一步，Hidost不仅考虑了文件的逻辑结构，而且还考虑了其​​内容，使得对SWF等区别度较小的格式具有更高的精度。

为了演示Hidost的出色检测性能，我们通过实验评估了两种格式：PDF和SWF。我们的评估协议旨在模拟数据驱动的检测方法的实际部署，并考虑到恶意数据的自然演变。在我们的协议中，检测模型在固定大小的数据窗口上进行训练，并部署一段时间。一旦模型被认为太旧，就会在另一个更新近的数据窗口上重新训练，并再次在有限的时间段内进行评估。与机器学习算法评估中常用的经典交叉验证方法不同，我们的实验性协议考虑了安全应用程序中数据的时间性质，并且从未预测过去的数据。

总之，本文的主要贡献如下：

•     基于静态机器学习的恶意软件检测器Hidost，这是第一个基于其逻辑结构和内容适用于不同文件格式的系统。

•     针对PDF和SWF两种格式的Hidost进行了实验性评估，旨在反映恶意软件检测器的操作环境，该软件在440,000个PDF文件和前所未有的40,000个SWF文件的数据集上执行。在我们的评估中，Hidost在PDF上的性能优于VirusTotal上的所有防病毒引擎，并在SWF文件中名列前茅。

•     用于两种文件格式（PDF和SWF）的Hidost原型实现，作为开源软件发布。

•     复制这项工作所需的源代码，包括作为开源软件发布的实验和绘图。

•     复制本作品所需的数据集。

**1.3      大纲**

本文的结构如下。Hidost适用的文件格式（即分层结构的文件格式）将在第2节中进行介绍，同时还将详细介绍PDF和SWF的逻辑结构。第3节介绍了Hidost的系统设计，其中包括从PDF和SWF格式中提取结构元素，特征定义，选择和压缩以及学习和分类。第4节介绍了实验评估，包括数据集和实验协议的描述以及对结果的讨论。我们讨论Hidost对其他文件格式的扩展，并为其应用程序的概念设计提供了办公文件格式OOXML和ODF最后，第6节给出了结论，并概述了未来工作的未决问题。

**2        分层结构的文件格式**

文件格式是作为存储某些信息的物理表示的手段而开发的。某些格式（例如文本文件）没有任何逻辑结构，但其他格式（例如HTML）却可以。HTML文件是HTML *元素*之间逻辑关系的物理表示。如图2中的例子所示，在一个HTML文件中，一个*p*元素可能是*body*元素的后代，而*body*元素又有*html*元素作为其父元素。

HTML元素具有层次结构形式的逻辑结构。本文中介绍的工作涉及以*分层结构化文件格式*检测恶意软件。文件格式的物理布局可能会大大偏离其逻辑布局，与所提出的方法的操作无关。分层结构的文件格式的例子包括：

•     可移植文档格式（PDF）

•     SWF文件格式（SWF）

•     可扩展标记语言（XML）

•     超文本标记语言（HTML）

•     开放文档格式（ODF），一种基于XML的办公文档格式

•     Office Open XML（OOXML），一种用于办公文档的基于XML的不同格式

•     可缩放矢量图形（SVG），一种基于XML的矢量图形格式

在下文中，我们描述了在Hidost，PDF和SWF中实现的两种文件格式的层次逻辑结构。

**2.1可移植文档格式（PDF）**本节从[26]复制（与适应），由互联网协会版权所有。

*可移植文档格式*（PDF）是一个开放标准，发布为ISO 32000-1：2008 [9]。PDF的语法包含以下四个主要元素：

•     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有效存储文件内容的各个部分。

PDF中有九种基本对象类型。简单的对象类型是布尔值，数字，字符串和空值。PDF字符串具有有限长度，并被括在括号“ （”和“ ）”中。Name类型用作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*（0 × 00 ）以外的任意字符。上述五种对象类型将被称为*原始的*类型在本文中。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。字典是在符号“ << ”和“ >> ” 之间包含的一组无用的键值对。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Stream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。最后，一个间接对象是以前定义的任何一个对象，这些对象提供了一个唯一的对象标识符，并包含在关键字obj 和endobj中。由于其独特的标识符，间接对象可以通过*间接引用*从其他对象*引用*。

PDF对象的语法在图3所示的简化的示例性PDF文件中示出。它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 以及obj 和endobj 关键字。这些对象是字典，因为它们被符号“ << ”和“ >> ” 包围。第一个是*Catalog*字典，由其*Type类型*条目表示，其中包含一个带有值*Catalog*的PDF名称。该目录有两个额外的字典条目：*页面*和*OpenAction*。*OpenAction*是嵌套字典的一个例子。它有两个条目：*S*，表示这是一个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*，表示文档中有两个页面，一个数组*Kids*可以用方括号标识，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。注意，每个Page对象都包含一个向后的对象

在*Parent*条目中引用Pages对象。总而言之，有三个引用指向相同的间接对象，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*。文档结构中的节点本身就是对象，边对应于子对象驻留在父对象下的名称。对于数组，父亲关系是无名称的，并且对应于各个元素的整数索引。严格地说，文档结构不是树，而是一个定向的有根循环图，因为间接引用可能指向文档结构中任何位置的其他对象。这个图可以简化为一棵合适的树，称为 *结构树*，如3.4节将详细阐述的那样，因此我们将限制自己以简化的树形式考虑PDF文档结构，如图4所示。

文档结构中的根节点是一个特殊的PDF字典，其中包含名称*Catalog*的强制*Type类型*条目。任何基本类型的对象都构成文档结构中的一个叶子，即终端节点。

我们在PDF结构树中定义一条*路径*，作为从目录字典开始并以基本类型的对象结束的一系列边。例如，在图4中，存在从根（即，最左边的节点）到名为/ Pages 和/ Count 的边缘到值为2的终端节点的路径。这个PDF文档结构中的路径定义我们表示*PDF结构路径*，在我们的方法中起着核心作用。我们将路径打印为从根节点开始到叶节点结束时在路径遍历期间遇到的所有边标签的序列。我们之前例子的路径将被打印为/ Pages / Count 。

以下列表显示了来自真实世界良性PDF文件的示例性结构路径：

/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

我们的调查显示，这些结构路径在文件中的存在最能表明文件是良性的，或者缺少文件表明文件是恶意的。例如，恶意文件不太可能包含元数据以最小化文件大小，当文档打开时它们不会跳转到文档中的页面，并且格式不正确，因此它们缺少路径，例如

/类型和/ Pages / Count 。

以下是来自真实世界恶意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，在打开文档并利用Adobe XML Forms Architecture（XFA）表单时，恶意文件倾向于执行存储在多个不同位置的JavaScript，因为恶意代码也可以从那里启动。

**2.2      SWF文件格式**

*SWF文件格式*（*SWF*，发音为*swiff*）是一种专有的二进制文件格式，其规范在[27]中发布。SWF文件包含一个标题和一系列标记，即具有预定义字段值的数据结构。指定了65种不同类型的标签，每种标签都定义了自己的一组具有不同名称和数据类型的字段。一些基本的SWF数据类型是[27]：

•     8位，16位，32位和64位整数，包括带符号和无符号的数组，这些类型和整数的可变字节数

•     不同宽度和精度的固定点和浮点数

•     宽度不是2的整数和定点数字

•     字符串

•     数据结构，如24位和32位彩色记录，矩形记录，2D转换矩阵等

图5显示了一个用于说明目的的非常小的SWF文件。显然，SWF的物理布局对于直接解释来说太模糊了。相反，我们对SWF逻辑结构的描述是基于解码的，人类可读的相同文件的描述，如图6所示。原始SWF文件的文本描述

是使用SWFRETools工具包[28]中的ConsoleDumper 类生成的，该工具包是用于反向工程SWF文件的开源Java工具包。

插图跳过文件头，因为它没有在我们的方法中使用。它显示了5个由虚线分开的SWF标签：2个*SetBackgroundColor*在字节代码0 × 14和0 × 1B 中，两个*ShowFrame*在字节代码0 × 19 和0 × 20 和一个*终结*在字节标记0 × 22 。SWF文件的标签按顺序排列。每个标签都有一个带有无符号16位小尾数*TagCodeAndLength*的标头字段，其包括10字节标签类型标识符和6-b标签长度字段，对于长于62B的标签，可选宽字段字段。

该文件中的第一个标签用于设置显示器的背景颜色。它是一个简单的标签，定义了红色（0 × AA = 170），绿色（0 × BB = 187）和蓝色（0 × CC = 204）颜色分量的三个无符号1-B值。第二个标签在一帧的持续时间内使画布的内容呈现在屏幕上。在此之后，背景颜色设置为＃112233，屏幕再次刷新一次。最后一个标记表示文件的结尾。

图7展示了我们的示例SWF文件的逻辑视图，其中文件构建为树形结构。它紧跟在图6的解码的SWF文件中呈现的布局。每个标签由树节点表示并且是抽象根节点的直接后代。从根到标记节点的边缘由标签类型名称标记的，在我们的情况下SetBackgroundColor ，ShowFrame，并结束。标签节点的后代是它的标题和字段。标题与简单标记为标题的边连接。通向字段的边缘用它们的名称标记，例如BackgroundColor 。标签字段的值表示为叶子，例如，第一个*红色*字段的值*SetBackgroundColor*标记，170。

我们定义了一个*路径*在SWF结构树，类似于在PDF结构树中的路径，为一系列的抽象根节点开始，并在叶节点结束的边缘。例如，从根节点到标记为End ，Header 和TagAndLength 的边的一条路径在叶节点中以值0结束。为了更好的可读性和与PDF的一致性，我们将正斜杠符号“ / ”指定为每个打印路径时的边缘标签; 因此，有问题的路径打印为/ End / Header / TagAndLength 。

在下面的章节中，我们将介绍如何处理PDF和SWF文件的逻辑结构以供学习算法使用，并描述Hidost的系统设计。

**3       系统设计**

Hidost被设计为恶意软件检测系统，能够学习根据逻辑结构区分恶意文件和良性文件。由于各种文件格式的语义异质性，很难想象单一格式可以作为所有可能的分层结构文件格式的“共同标准”。然而，我们的设计明确地将格式特定的处理步骤与检测方法分开。因此，我们目前在PDF和SWF格式中测试的方法可以通过实现特定于格式的组件来扩展到其他格式，而无需重新构建其通用框架。所提出的方法是作为研究原型实现的，其特征提取子系统被公开为开源软件[29]。已发布的代码包含一个用于从PDF（用C ++实现）和SWF文件（用Python和Java实现）提取功能的工具集。实验复制代码将单独发布，如第4节所述。

Hidost的系统设计如图8所示.Hidost有六个主要阶段：结构提取，结构路径合并，特征选择，矢量化，学习和分类。结构抽取将特定格式的结构特征转换为表示结构层次结构中的路径的通用数据结构 - 结构化多图。结构路径合并旨在将结构路径转换为更一般的形式，消除工件。特征选择涉及找到成功的机器学习应用程序所需的最小特征集。矢量化将结构化多图表转换为机器学习方法处理的数字矢量。学习根据编码在特征向量中的属性生成恶意和良性文件的区分模型。

在以下小节中，我们将详细介绍我们方法的主要阶段。

**3.1       文件结构提取**

我们方法的第一步是将文件转换为更抽象的表示形式，即逻辑结构。这一步对我们的方法至关重要，因为它实现了两个关键目标：（a）使用逻辑结构区分恶意文件和良性文件，以及（b）适用于多种文件格式。

分层结构文件格式的逻辑结构的适当表示是*结构化多图*。多图是普通地图数据结构的一般化，也称为字典或关联数组。尽管映射提供了键与对应值之间的映射，但multimaps将键映射到一*组值*。结构化multimap是一个multimap，它将结构树的每个结构路径映射到位于给定路径上的所有叶子集合。在地图术语中，结构路径表示路径映射以表示*值*的所有叶子的*键*和集合的地图。结构化多图的一个例子如图9所示。同一个键的多个值使用竖线符号“ | ”分隔 ”。Hidost使用本节后面介绍的简化形式的结构化多图。

逻辑文件结构的本质需要一个multimap而不是一个map，因为多个叶子可以通过单个结构路径到达。对于PDF，当路径包含具有多个元素的数组时，会发生这种情况，例如，路径/ Pages / Kids / MediaBox 包含两个数组并达到八个叶子。在SWF文件的情况下，除了数组之外，同一类型的多个标签会导致多个叶子位于相同的路径中。

以下两节介绍了PDF和SWF结构提取的实现。

***3.1.1     PDF***

PDF逻辑结构被组织为定向的有根循环图。为了将其转换为结构化多图，首先需要将图减少为定向的有根树。这是通过从图中删除所有循环来实现的。当来自深度*d R*处的树节点的间接引用指向位于深度*d T*< *d R*处的相同路径上的树节点时，PDF逻辑结构中存在周期。图3显示了一个示例：深度为3 的路径/ Pages / Kids / Parent 上的间接引用指向位于/ Pages 处的字典深度为1，位于同一条路径上。间接引用也可能导致提取的树结构不一致。当位于不同路径的两个间接引用引用同一个对象时，它是不明确的，这是“真实”的。

通过基于广度优先搜索（BFS）算法的简单程序隐式解决了所描述的两个问题 - 结构图中的循环和多个引用。需要一个强大的PDF解析器来导航混淆或格式不正确的PDF文件的PDF结构。Hidost利用开源的Poppler PDF渲染库[30]，版本为0.18.4。该过程首先查找文件结构的根节点，即*目录*字典。然后，它在整个文件结构图上执行广度优先搜索，插入所有对（*p*，*l*），其中*p*是结构路径，*l*是位于路径*p*上的叶节点，结果到结构化多图中。通过跳过指向先前访问的对象的间接引用并将它们视为叶节点来避免循环。按照字母顺序的子节点枚举确保多个引用在同一节点上的一致解析，以便每次遍历PDF文件的结构图都会生成相同的结构化多图。

更确切地说，Hidost并没有使用完整的结构化多图，而是使用了简化。这种简化涉及非数字数据类型的处理，即除整数，实数和布尔值之外的所有类型。字符串，PDF名称和其他非数字类型，都可转换为字符串，用常数值1替换。非数值的二值化可以通过比较图1和图2来看出。9和10.这种治疗的选择是在完全丢弃非数字值和进行广泛评估之间的一种折衷。在相关工作中，已经提出了不同的处理字符串型数据类型的方法，从静态的，例如，在度量空间中嵌入字符串[10,31]，用简单的属性来表征它们，例如它们的长度，熵或关键字分布[11，12]或测试它们以获得有效的CPU指令[15]，以动态执行，例如对字符串[14,21]的CPU模拟或其执行[19,20,22]。然而，所有这些方法要么与所需的静态系统设计（动态评估）相冲突，要么降低计算性能（字符串嵌入和CPU指令测试），要么很容易被忽略（简单字符串属性）。

实验证实了使用二值化非数字值与完全省略相比的积极影响。第4.3.4节评估了从SL2013使用的纯二元特征迁移到Hidost数值的影响。

***3.1.2     SWF***

SWF逻辑结构比PDF更直接，因为它不包含循环或含糊不清。在Hidost中实现的方法首先使用ConsoleDumper SWFRETools工具包的类来解析SWF文件并产生其文本描述，如图6所示。解析此工具生成的输出足以提取SWF逻辑结构。文本输出的每一行以零个或多个空格开头，后跟一个开方括号“[”包含代表结构树中一个边的标记或字段名称。该边与根节点之间的距离被编码为括号之前的空格数，除以2.因此，带有括号前面没有空格的行表示新标签的开始。通过跟踪最近分析的边缘

在层次结构中的每个级别，都可以重建当前行中到边缘的整个路径。最后，如果边名继承冒号，则此边表示一个标记字段，并且行的其余部分对该字段的值进行编码，即结构树中的一个叶。对（*p*，*v*），其中*p*是当前行的路径，*v*是解析值，然后插入到结构化多图中。字符串和其他非数字类型以与PDF相同的方式进行二进制化。

与图6的SWFRETools输出相对应的结构多图在图11的左侧示出。以下部分描述了我们的方法的第二个处理步骤，即结构路径合并。

**3.2        结构路径合并**

许多文件格式的语法丰富性和灵活性使得语义等价但语法上不同的结构成为可能。这种语法多态性可能会降低检测的准确性，并且还会使攻击者完全避免检测特定文件。为了解决这个问题，我们开发了一种启发式技术，用于合并结构路径，从而将多态路径缩减为一致的表示。这种技术可以作为PDF格式的最佳例证。

我们已经观察到许多共同的路径

PDF文件显示与其他路径结构相似。事实上，我们能够识别彼此相似但不完全相同的路径组。这些群体中的路径以两种方式之一表现出相似性。除了一个可自定义的路径组件外，某些组中的路径是相同的，而其他组中的路径共享一个共同的重复子路径。然而，重要的是，一组相似路径中的所有路径指的是PDF中具有相同目的的对象，即相同的*语义含义*。例如，路径/ Pages / Kids / Resources 和/ Pages / Kids / Kids / Resources 有一个共同的重复子路径，/ Kids，但都是指具有相同目的的PDF字典 - 为呈现PDF文件页面所需的资源提供名称。在语义上，结构提取算法在访问页面的资源字典 - 所有资源之前采用哪条路径是无关紧要的

字典具有相同的语义含义。同样，路径/ Pages / Kids / Resources / Font / F1 和/ Pages / Kids / Resources / Font / F42 仅在最后一个路径段中有所不同，PDF标准要求用户定义这两个路径段，但都涉及字体词典描述用于PDF文件的字体。同样，无论具体的PDF编写器赋予字体字典具体的名称，所有的字体字典在语义上都是相同的。

语义上等价的路径组存在着对SL2013特征定义的效用的问题，SL2013特征定义将每条路径视为一个独特的特征。通过将所有语义等同的路径合并到一个特征来保留路径的语义更有意义。这种被称为*结构路径整合*（SPC）的思想是在Hidost中实现的，并在4.3.3节中进行了实验评估。

SPC的实现基于使用正则表达式替换关键路径组件。重复的子路径将从路径中完全删除。例如，以上作为具有共同重复子路径的示例所指示的两个路径将被合并到路径/页面/资源中，去除重复子路径/孩子。另一方面，用户定义的路径组件是*匿名的*，即用占位符路径组件/名称替换。例如，上面例子中用户定义字体名称的两个路径将合并到路径/ Pages / Kids / Resources / Name中（如果关于重复路径的规则当然没有事先应用）。表1列出了SPC规则Hidost用于PDF，使用B实施OOST .R EGEX 库。每个规则包含两个正则表达式：一个用于搜索要替换的模式（左），另一个用于确定替换字符串（右）。

单个合并规则可以应用于多组语义等价路径。例如，路径/ Pages / Kids / Resources 和/ Pages / Kids / MediaBox 可以通过相同的规则进行合并，但生成的路径/ Pages / Resources和/ Pages / MediaBox 仍属于不同的组，因此，两个不同的功能。

表1中的SPC规则是对我们的数据集中出现的结构路径进行实证研究的结果，其目的是最小化其转换后的总数。但是，分析仅限于捕获PDF文档结构的通用分支而不是数据集特定工件的规则。表1中一些可识别的分支包括匿名项目，如资源（1和13），各种名称树（全局（5）和结构树（6））的条目，用于将自定义名称映射到其他对象（9）的字典，色彩空间项目（17和18）或嵌入文件的名称（19）。其他规则被用来压扁层次结构（2,3,7,8和10）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **表1** PDF结构路径合并规则   |  |  | | --- | --- | | 搜索正则表达式 | 替换正则表达式 | | 1.      / Resources /（ExtGState | ColorSpace | Pattern | Shading |  x对象|字体|属性|帕拉）/ [/] + | / Resources / \ 1 / Name | | 2.     Pages /（Kids / | Parent /）\*（Kids $ | Kids / | Parent / | Parent $） | 页/ | | 3.      /（Kids / | Parent /）\*（Kids $ | Kids / | Parent / | Parent $） | / | | 4.       （Prev / | Next / | First / | Last /）+ | <空字符串> | | 5.      名称/（Dests | AP | JavaScript |页面|模板| IDS |  URLS | EmbeddedFiles | AlternatePresentations |  译丛）/（儿童/ |家长/）\*名称 | 名称/ \ 1 /名称 | | 6.      StructTreeRoot / IDTree /（Kids /）\*名称 | StructTreeRoot / IDTree /名称 | | 7.      （StructTreeRoot / ParentTree | PageLabels）/（Kids / |  家长/）+（订购数量|限制） | \ 1 \ \ 3 | | 8.      StructTreeRoot / ParentTree / Nums /（K / | P /）+ | StructTreeRoot / ParentTree /订购数量/ | | 9.      （StructTreeRoot | Outlines / SE）/（RoleMap |  类映射）/ [/] + | \ 1 / \ 2 /名称 | | 10.     （StructTreeRoot | Outline / SE）/（K / | P /）\* | \ 1 / | | 11.      （Extensions | Dests）/ [/] + | \ 1 /名称 | | 12.     字体/（[/] +）/ CharProcs / [/] + | 字体/ \ 1 / CharProcs /名称 | | 13.     （AcroForm /（Fields / | C0 /）？DR /）（ExtGState | ColorSpace |  图案|底纹| x对象|字体|属性）/ [/] + | \ 1 \ 3 /名称 | | 14.      / AP /（D | N）/ [/] + | / AP / \ 1 /名称 | | 15.      线程/ F /（V / | N /）\* | 线程/ F | | 16.     （StructTreeRoot | Outlines / SE）/ Info / [/] + | \ 1 /信息/名称 | | 17.     ColorSpace /（[/] +）/着色剂/ [/] + | ColorSpace / \ 1 /着色剂/名称 | | 18.      ColorSpace / Colorants / [/] + | 的ColorSpace /着色剂/名称 | | 19.      收藏/架构/ [/] + | 收集/模式/名称 | |

并将链接列表转换为浅层集合（4），以便在同一层次上创建其元素的通用统一视图。我们将读者引用到PDF标准[9]中，详细解释PDF文档结构的这些分支。

由于SWF逻辑文件结构相对较浅，并且禁止用户定义的路径组件，因此只有两种SPC规则被编译用于表2中列出的这种格式，这两种规则都用于处理重复的子路径。

没有试图编译完整的SPC规则列表。特别是对PDF而言，我们有充足的机会进一步匿名化和层级化，例如我们的规则中未涉及的名称树和数字树。通常，为了扩展列表，建议阅读PDF标准，查找允许用户定义名称的PDF文档结构中的位置，或者在那里

**表2** SWF结构路径合并规则

|  |  |
| --- | --- |
| 搜索正则表达式 | 替代正则表达式 |
| （DefineSprite / ControlTags /）{2，} | DefineSprite /  ControlTags / |
| （符号/名称/）{2，} | 符号/名称/ |

是一个明确定义的列表或层次结构。然而，与SL2013相比，即使这套有限的规则也提供了以下关键优势：

•     减少攻击面。没有SPC，出现次数高于阈值的每条不同路径都构成一个特征。在这种情况下，一个努力逃避检测的攻击者可能通过隐藏一个不同于该特征集中任何一个的自定义路径的恶意有效载荷来执行隐藏攻击。例如，具有长随机生成名称的字体的路径很可能以前不会遇到。插入其中的恶意有效载荷对于其特征集中没有该特定路径的检测器是不可见的。对于不允许用户定义的路径的SWF，有效载荷可能隐藏在非常深的层次结构中，而在“普通”文件中不会遇到。PDF也受此漏洞的影响。隐藏攻击实施便宜，是SL2013的主要弱点。

•     有限的功能设置漂移时间。在现实世界的机器学习应用中，手头的问题经常会随时间而改变。在安全应用中尤其如此，在这种应用中，防御者被迫适应不可预测的攻击变化。这个问题在机器学习中被称为概念漂移[32]，并且最近开始引起对安全文献的兴趣[33]。

随着时间的推移，数据的不断变化使得分类器越来越过时。因此，需要定期更新学习模型，即定期分类器再训练。在这项工作中，与数据相关的特性，建议在每次再培训前重新进行特征选择，以便更好地适应概念漂移。定期特征选择会导致现有特征的过时以及在两个再训练期之间添加新特征。我们将周期性特征选择引起的特征集更改称为featuresetdrift。由于结构路径定义的灵活性，PDF比SWF更容易受到特征集漂移的影响。如4.3.3节所示，SPC被有效地用于减少Hidost中的特征集漂移。

•     特征空间降维。最后，SPC对功能的总数有巨大的影响。特征空间维度直接影响学习算法的运行时间和内存需求。在我们的定期再训练的PDF实验中，平均特征集大小减少了88％，从每次训练的10,412.5到1237.4个特征。但是，对于手动制作的路径，SPC的有效性存在限制。由于它不知道语义上有效的路径，所以SPC无法处理不可预见的情况，例如目录字典中的任意名称。为了解决PDF逻辑结构覆盖范围中的这个最后的“盲点”，需要采用白名单方法，其完整和最新的表示方式 - 整个结构的模型 - 这超出了本文的范围。

减少的攻击面和有限的特征集漂移对Hidost作为基于机器学习的检测器的操作安全性起着重要作用。大量减少的功能数量使其能够应用于更大的数据集。所描述的改进结合在一起，使Hidost在真实世界的操作环境中作为准确，可靠和安全的恶意文件检测器的适用性迈出了一大步。

**3.3      功能选择**

尽管通过结构路径合并减少了语法多态性，但观察数据中可能仍然存在很少发生的路径。使用这样的路径来构建区分模型会增加输入空间的维度而不会提高分类准确性。因此，必须执行特征选择（在其他机器学习应用中常见），以限制罕见特征的影响。在介绍具体特征选择技术之前，我们讨论了本文详细研究的两种格式中罕见特征出现的原因。

SWF文件格式规范[27]严格定义所有标签及其所有字段的名称，禁止定制。因此，Hidost的SWF功能理论上包含由SWF规范定义的每个结构路径。但是，实际上，我们没有努力列举SWF逻辑结构中的所有路径。相反，特征集合包括在训练数据集中*观察到的*所有路径，总共3177个。

相比之下，PDF文件格式规范[9]允许在任何PDF字典中使用用户定义的名称，实质上启用无限数量的不同路径。我们的数据表明，这个PDF功能在实践中被广泛使用，因为我们已经观察到超过900万个不同的PDF结构路径。但是，这些路径中不会出现在多个文件中。发生在小数据集中的这些和其他路径被认为是异常的。因此，原SL2013方法选择了出现在固定数量的训练文件中的PDF路径，即其特征集合，即1000。该阈值控制检测精度（更多路径）和模型简单性（更少路径）之间的折衷，并且可以自由调整。

在SPC之后并且在我们的定期再训练实验方案中的每次训练之前，我们应用相同的出现阈值，即1000个文件，其对应于训练集大小的约1％。这与SL2013形成鲜明对比，SL2013在“事后”中进行特征选择，对整个数据集进行一次特征选择。

**3.4      矢量化**

结构化多图是PDF和SWF逻辑文件结构的合适表示，但它们不能直接用于机器学习算法。它们首先需要在一个称为向量化的过程中转换成特征向量，即*特征空间* R *N*中的点。

在矢量化过程中，结构化多图表首先被结构图 - 普通地图数据结构所替代，这些结构图将结构路径映射到相应的单个数值。为此，对应于多映射中的一个结构路径的每组值被减少到其中值。我们选择中位数作为比均值更强的统计量（这里，我们在统计学意义上使用*鲁棒性*这一术语，表示中位数在存在异常值的情况下提供了对该组值的更好表征，而不是它提供了针对对抗规避的任何稳健性）。这条规则的唯一例外是，主要由布尔值组成的SWF结构化多元图中的值集合被简化为它们的意义，而不是中值。均值保留有关布尔值的更多信息，而不是中值，它只能为0，，或1，并且不存在异常值。由于其逻辑结构具有相对较少的布尔值，因此该例外不适用于PDF。

构造图被转换成特征矢量**˚F** ∈R *Ñ*通过保留一个单独的维度为每个特定结构的路径和使用从构造图值作为具体的尺寸的值。在特征提取之前和特征选择期间定义个体结构路径到特征向量维的映射，并且在训练和分类之前均匀地应用于所有结构图。因此，特定的结构路径对应于每个特征向量中的相同维度，使得学习算法能够理解特征向量。

学习算法使用的所有特征的有序集合是其*特征集合*。图10和11分别示出了从PDF和SWF结构多图获得的特征向量。当特征集与结构化多重映射中的一组键相同并且分配了特征向量的每个值时，它们显示一个简单情况。然而，在实践中，文件通常不包含特征集中存在的所有结构路径，并且特征向量中的相应值被设置为零。总之，对应于结构化多映射*m*的特征向量是点**f** = *f*1，*f*2，... ，*f N*在特征空间R *N中*，特定值定义为

*˚F 我* =         中位数（*米* [ *p 我* ] ），*p 我* Element; *米*∀ *我* ∈ 1 ，*Ñ*（1）

                  0，                            否则

在此，*p 我*表示*我*个在功能设置和路径*米* [ *p 我* ]表示在多重映射值*米*与该路径相关联。

**3.5       学习和分类**

到目前为止所呈现的阶段将样本（即，文件）转换成适合作为机器学习算法的输入的特征向量。具体机器学习分类器的选择取决于多个参数，例如，数据集大小，特征空间维度，可用计算资源，针对对抗攻击的鲁棒性等，并且分类器针对不同用途而量身定制。因此，已公布的Hidost实施不包括学习和分类子系统。相反，其输出可以与读者选择的分类器一起使用。对于本文中介绍的实验，利用开源的scikit-learn Python机器学习库[34]的版本0.15.0b2的随机森林实现。作为实验性复制代码的一部分，Hidost的这部分单独发布，

随机森林[35]是一个整体分类器。它是通过使用CART方法生长决策树的森林来培训的。*t个*RF树中的每一个都是在它自己的固定尺寸随机子集上随着替换而绘制的训练数据。在训练期间在树的每个分支处，提供最优分割的特征从包含*f*RF的随机子集中选择以前不用于此树的功能。在分类过程中，每棵树的决策都被计为一票，总体结果是大多数选票。随机森林以其卓越的泛化能力和抗数据噪声的稳健性而闻名。对于实验评估，将森林大小设置为200棵树，并将其他所有参数设置为scikit-learn默认值。

**4       实验评估**

进行了广泛的实验评估，以评估Hidost的检测性能并将其与相关工作进行比较。重现所有实验和绘图所需的全部源代码和数据集已经作为开源软件发布[36]。

**4.1       实验数据集**

实验在两个数据集上运行，每个文件格式一个。两者都是从VirusTotal [37]收集的，该网站使用许多防病毒引擎对因特网用户上传的文件进行分析。VirusTotal为研究人员提供检测结果，使我们能够将Hidost的检测性能与部署的防病毒引擎的检测性能进行比较。在我们的实验中，

对那些被恶意标记为*至少有5个*防病毒引擎恶意的文件以及那些被*所有*防病毒引擎标记为良性的良性文件。由于其真实类别标签的高度不确定性，其余文件（由1到4个防病毒引擎标记为恶意文件）将从实验中丢弃。

我们的PDF数据库包含439,563（446 GiB）文件，407,037（443 GiB）良性和32,567（2.7 GiB）恶意文件。它们是在2012年7月16日至10月21日的14周内收集的。这与[26]中*10Weeks*实验所用的数据集相同，可以与该工作进行直接比较。

SWF数据集在2013年8月1日至2014年3月8日期间收集，包含40,816（14.2 GiB）个文件，38,326（14.1 GiB）个良性和2490（190 MiB）恶意文件。在收集期间，VirusTotal SWF数据具有52：1左右的良性比例; 因此，对良性数据进行随机子采样以使其与PDF数据的比率大致匹配。

**4.2       实验协议**

我们的实验协议有以下两个主要目标：（a）在实际条件下评估Hidost的性能;（b）将Hidost在PDF上的检测性能与其前身SL2013进行比较。为此，我们采用同一出版物的*10Weeks*实验的实验方案。

该*10Weeks*实验试图接近暴露在变化，及时调整自己的攻击的恶意软件检测系统的真实世界，作战环境。为此，它采用定期再培训和评估。训练和评估数据集以滑动窗口方式进行组合，即对于评估数据的每一周，分类器在前4周的训练数据上进行训练。

为了遵循*10Weeks*的协议试验尽可能接近，两个实验数据集按如下分区。收集文件的时间段被分为14个较小的连续时间段。对于PDF，每个时间段只有一周，SWF为15天，最后一个为25天。每个时间段都被分配一个桶，并且根据首次出现的时间段将每个文件放入一个桶中。然后，应用滑动窗口方法，将四个连续的桶连接到训练数据集中，并使用下面的桶作为相应的评估数据集，从而得到10个数据分区用于定期再训练。在每次再培训之前，即每周对于PDF，每15天SWF，特征提取的所有四个步骤（即结构提取，SPC，特征选择，

因此，生成的PDF文件分区与[26]中使用的相同，并且SWF数据集遵循相同的设计。数据集在图1和2中示出。12,13和14.每个再培训事件都标有培训日期，即标志评估期间数据收集开始的那一天。

尽管PDF的良恶性分级比率在所有时间段内大致相等，但恶意和良性SWF文件在时间上的分布高度倾斜，如图5和图5所示。在第一次评估期前收集了多达70％的SWF-Normal数据集中的恶意SWF文件，而在第五次评估期之前，只有不到10％的良性SWF文件出现。其结果是在大多数培训和评估数据集中出现高水平的失衡。为了量化高级别不平衡对检测性能的影响，我们为SWF数据生成了另一个数据分区，标记为SWF-KeepMal，如图14所示，其中不会丢弃超过四个周期的恶意训练样本。相反，它们用于后续所有时期的培训。

**4.3      实验结果**

在PDF和SWF数据上运行的不同方法的实验结果如图1和2所示。15和16分别。这些方法在典型的分类任务的四个性能指标中进行比较：真实（TPR）和假阳性率（FPR），准确性和受试者工作特征下的面积（AUROC）。AUROC与精度 - 召回曲线下的面积类似，对于平衡和不平衡数据集都是一个很好的检测性能指标。由于算法的随机性，对所有随机森林实验绘制了10次独立运行的平均值。这些实验的方差由于其非常低的价值而从图中省略。SL2013采用支持向量机（SVM）分类器，一种确定性算法; 因此，它的结果是从一次实验中获得的。

|  |
| --- |
|  |
|  |  |

图15显示了PDF数据中Hidost和SL2013的不同变体的结果。随着SL2013的准确再现，使用随机森林分类器而不是SVM评估相同的方法。Hidost显示了二进制和数字特征。在SWF数据集SWF-Normal和SWF-KeepMal上，这两个Hidost变体也显示在图16中。

***4.3.1      分类表现***

图15显示了PDF数据集上Hidost与SL2013的直接比较。引入了SL2013的随机森林变体，可以独立比较两种方法的特征集和分类器。可以看出，SL2013的结果，尤其是10月1日的真实阳性率，可以通过在相同的二元松散特征上使用随机森林而不是SVM来快速提高。另一方面，对于包括SVM在内的所有方法，AUROC指示的预期分类性能实际上是相等的。AUROC在给定时间内的任何两种方法之间的最大差异仅为0.006，并且在每个时期所有方法的AUROC均在0.99以上。可以得出结论，Hidost在PDF数据上实现了其前身SL2013的出色分类性能。

Hidost在SWF数据上的表现与其在PDF上的成功并不相同。虽然平均检测精度在95％以上，如图16所示，但由于SWF数据集中良性样本的15：1大类不平衡，因此精度并不是有意义的性能指标。类别失衡在个别时间段的数据集中甚至更大，如图1和图2所示。13和14，尤其是在SWF-Normal的情况下。

阶级失衡的影响清楚地反映在结果中。在SWF-KeepMal上，随着时间的推移累积恶意训练样本，Hidost的总体AUROC要高于SWF-Normal，其中恶意软件会在四个时间段后丢弃。SWF-KeepMal在早期阶段的真实阳性率比较平衡时，比SWF-Normal高5％至10％。从12月29日开始，在阶级失衡急剧增加之后，优势跃升至20％左右，这是一个巨大的进步。获得更多的恶意培训数据也增加了误报率，但具有数字特征的变体的增加保持在界限内，除了最后一段时间。这些发现清楚地表明了Hidost进一步提高检测性能的潜力，给予恶意SWF训练数据更大的可用性。但是，由于SWF数据集仅包含2490个恶意样本，因此无法准确量化改进的可能性。

***4.3.2与防病毒引擎的比较***为了在日常的实际操作条件下获得Hidost检测性能的估计值，有必要从更广的角度来考虑。与防病毒引擎直接比较，大多数人为了安全而广泛使用的恶意软件检测器提供了这样一种现实检查。我们在实验过程中比较了检测器的真正阳性计数，即他们正确标记的恶意样本的数量。根据我们的基本事实的定义，样本

至多四个防病毒引擎被标记为恶意软件被过滤掉。因此，防病毒引擎没有误报，无法在这个意义上进行比较。

图17显示了Hidost（平均10次实验运行）和VirusTotal在PDF和SWF文件上部署的防病毒引擎的结果。在实验结束*后*收集防病毒检测结果，而不是在每个新文件提交给VirusTotal后立即收集。这为反病毒引擎提供了在此期间更新其检测机制的机会，并正确检测到任何文件在其首次提交和收集检测结果之间重新提交的文件。

尽管如此，Hidost是最好的整体。它的PDF检测率是无与伦比的，即使是SWF真正的阳性计数，与“在纸上”的PDF相比，差很多，与现实条件下的现有产品相比，是最好的。

***4.3.3      结构路径整合的影响***

SPC是Hidost在SL2013方面的主要创新之一; 因此，对其对系统性能的影响的评估只是合适的。图15表明，SPC不会影响检测性能，既不是积极也不是消极的。当使用相同的随机森林分类器时，SL2013（无SPC）的结果在PDF上与Hidost（使用SPC）的结果几乎相同。对SWF的影响可忽略不计，因为其严格的逻辑结构不允许用户定义的路径，从而导致对SPC的最低限度需求。

但是，SPC对特征集漂移具有强烈的积极作用。图18显示了我们的实验中使用PDF数据进行周期性再训练和周期性特征选择时的特征集漂移。可以观察到，在我们为期10周的实验的前半部分，特征集已经扩展，每周新特征的比例高达9％，而在下半年，许多特征被发现已经过时并且从特征中删除组。在第8周特征去除特别高，当不使用SPC时，前一周所有特征中几乎五分之一被删除。另一方面，当使用SPC时，在整个实验中连续星期的特征集之间的重叠远高于90％。总体而言，在Hidost中引入SPC可将特征集漂移减少约50％。

***4.3.4      数字特征的影响***

在Hidost中引入的另一个新颖之处在于使用数字而不是二进制特征，反映了从纯结构学习到结构学习与内容结合的转变。在这里，我们评估数字特征对性能的影响。

在PDF中，二进制和数字特征之间的差异是微不足道的，如图15所示。另一方面，对SWF的影响很大程度上是积极的。如图16所示，数值始终优于两个SWF数据集上的二进制特征。他们表现出对假阳性率最高的影响，将其降低多达50％。TPR和AUROC也显示出显着的整体改善。

两种文件格式的结果之间的差异的原因可能在于针对它们的攻击的性质。恶意PDF文件通常使用良性文件中罕见的功能，即它们的*结构*不同，而恶意SWF文件通常基于不同的值（即*内容*）在特定路径上进行攻击，尽管这些路径在良性文件中也很常见。尽管二进制特征足以描述逻辑结构，但数字特征增加的表达能力使得能够描述并因此检测基于结构攻击和基于内容的攻击。

**4.4 SL2013结果的再现** SL2013 [26]中发表的所有性能评估实验的结果使用原始RBF SVM准确复制，*C* = 12和γ= 0.0025。然而，当试图重现逃避鲁棒性评估实验时，我们发现了一个缺陷

针对RBF SVM的模仿攻击的源代码。也就是说，不是模仿良性组合中最良性的样本，而是将恶意组合中最良性的样本用作模仿目标。因此，所生成的攻击数据集被RBF SVM成功检测到。在发现这个缺陷后，我们将其删除并进行了一次纠正实验。这次攻击非常成功，准确率降至50％。因此，我们收回了[26]中发表的模仿攻击实验的结果，并将修正后的攻击的成功看作是对抗RBF SVM在敌对环境中鲁棒性的证据。随机森林的健壮性保证仍然是未来研究的主题。

**5      讨论**

Hidost引入的主要新颖之处在于它适用于多种文件格式，在PDF和SWF上实施并通过实验确认。它应用于其他分层结构的文件格式，例如XML，HTML，ODF，OOXML和SVG，需要现有解析器的工具或开发新的解析器，每种文件格式都需要一个。鉴于能够解析特定的文件格式，将其合并到Hidost中相当于开发结构提取模块。这一步必须专门针对每种文件格式。在下文中，我们讨论各种分层结构文件格式的文件结构和内容提取。

XML和相关的HTML和SVG具有非常清晰明确的层次结构，代表了这些格式的基石之一。例如，图2描述了一个具有路径/ html / body / p 的HTML文件。此外，还有许多用于XML文件的成熟的开源解析器。我们估计实现从XML文件中提取逻辑文档结构和内容非常简单。

尽管ODF和OOXML主要基于XML，但通常会将多个XML文件组合成ZIP文件，因此需要一些额外的处理。这两种格式都规定了一组文件和目录，其中内容，布局和元数据是分开组织的。格式区分文本和图形内容; 文本与逻辑结构一起存储在XML文件中，并以图形方式存储在目录层次结构中的单独文件中。我们观察到文件和目录本身是分层组织的，剩余的逻辑结构在XML文件中描述。以下显示了ODF文件中的简化文件和目录布局：

。

| - content.xml

| - manifest.rdf

| - META-INF

|     \ - manifest.xml

| - meta.xml

| - mimetype

| - settings.xml

| - styles.xml

\ - 缩略图

\ - thumbnail.png

我们认为目录层次结构是逻辑结构的最高层。其中，ZIP存档的根目录代表整个结构层次结构的根节点。可以将XML文件视为以根文件系统层次结构中相应节点为根的子树。例如，ODF规定图19中描述的文件meta.xml 位于根目录内，并具有一组描述文档元数据的XML标签。鉴于这种结构，*dc：creator*标签的路径是：

/meta.xml/office:document-meta/office:meta

→ / dc：创作者

通过将目录层次视为逻辑结构的顶层，将XML文件视为子树，我们确保完整且明确地提取逻辑结构。与PDF和SWF相比，我们将给定XML文件的文件系统路径（相对于ZIP存档的根目录）预先添加到从文件本身提取的结构路径。

ODF和OOXML存在多个解析器，其中一些解析器是开源的。我们相信，在适度的努力下，可以为这两种格式开发结构提取模块。此外，在许多情况下，完全良性的OOXML文件被用作嵌入恶意SWF文件的容器，Hidost已经可以处理它们。

结构化路径整合是Hidost中第二个也是最后一个格式特定的步骤，需要进行一些调整。我们期望不同的格式对SPC有不同的要求，并承认有必要更深入地理解SPC规则开发的文件格式。PDF与SWF的SPC规则的多样性证实了这一假设。

最后，Hidost对给定文件格式的适用性并不意味着它对其有效。例如，尽管我们坚信将Hidost扩展到XML是直截了当的，但它的有效性（以检测XML文件中伪装的恶意软件的能力来衡量）只能通过实验进行评估。但是，将其结构和内容用于建模使其更有可能取得成功。

**6      结论**

在本文中，我们介绍了Hidost，一种基于机器学习的恶意软件检测系统。它代表了先前发表的方法SL2013 [26]的扩展。Hidost是第一个基于静态机器学习的恶意软件检测器，旨在操作多种文件类型。它通过根据结构和内容制作恶意和良性样本模型来实现这一目标。

在一个实际的实验中对实际数据集进行了评估，这些实验包括多个月的定期再训练，Hidost胜过了网站VirusTotal部署的所有防病毒引擎，并检测到最高数量的恶意PDF文件。它也在SWF恶意软件中名列前茅。与其前身SL2013相比，它不太容易受到隐藏在PDF文件模糊部分的恶意软件的影响。通过定期再训练，Hidost也变得更加强大，能够不断适应恶意软件以更新防御。最后，其大大减少的特征集维度使其能够在非常大的数据集上高效地应用。

对于Hidost来说，合理的下一步是对其他分层结构文件格式的实施和评估。特别重要的是Microsoft Office使用的格式的应用程序，因为它们广泛用于最近的针对性攻击。本文提出了该应用的概念设计。展望未来，更高级的字符串处理方法的发展可能证明对于逻辑结构和数字内容不能提供足够区分能力的格式来检测恶意软件是不可或缺的。

**致谢**

我们承认Deutsche Forschungsgemeinschaft和Tübingen大学开放获取出版基金的支持。

**作者的贡献**

NS参与构思和设计研究，实施方法，参与数据收集，开展实验，起草了手稿。PL参与构思和设计研究，系统设计和数据收集，并帮助起草稿件。两位作者都阅读并批准了最终稿件。

**利益争夺**

作者声明他们没有竞争利益。

**作者的细节**

1认知系统，计算机科学系，美国

Tübingen，Sand 1，72076，德国蒂宾根。2慕尼黑办公室，欧洲研究中心，华为技术杜塞尔多夫有限公司，Riessstr。25 D-3.OG，80992München，德国。

收到日期：2015年11月29日接受日期：2016年9月7日

**参考**

1.      赛门铁克，2014年互联网安全威胁报告，第19卷（2014年）。[https://www.symantec.com/content/en/us/enterprise/other\_resources/ b-istr\_main\_report\_v19\_21291018.en-us.pdf。](https://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf)2015年4月13日访问

2.      思科2014年度安全报告（2014年）。[http://www.cisco.com/web/ offer / gist\_ty2\_asset / Cisco\_2014\_ASR.pdf。](http://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014_ASR.pdf)2015年4月13日访问

3.      Sophos，“2014年安全威胁报告”（2014年）。[http://www.sophos.com/en us / medialibrary / pdfs / other / sophos-security-threat-report-2014.pdf。](http://www.sophos.com/en-us/medialibrary/pdfs/other/sophos-security-threat-report-2014.pdf)2015年4月13日访问

4.      赛门铁克，2014年互联网安全威胁报告，第19卷，附录（2014年）。[https://www.symantec.com/content/en/us/enterprise/ other\_resources / b-istr\_appendices\_v19\_221284438.en-us.pdf。](https://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_appendices_v19_221284438.en-us.pdf)2015年4月13日访问

5.      赛门铁克，2015年互联网安全威胁报告，第20卷（2015年）。[http：// know.symantec.com/LP=1123。](http://know.symantec.com/LP=1123)2015年4月15日访问

6.      录制的未来，闪现：漏洞利用工具包使用的十大漏洞（2015年）。[https://www.recordedfuture.com/top-vulnerabilities-2015/。](https://www.recordedfuture.com/top-vulnerabilities-2015/)

2015年11月15日访问

7.      WJ Li，SJ Stolfo，A Stavrou，E Androulaki，AD Keromytis，*检测入侵和恶意软件及脆弱性评估（DIMVA）*。一项关于携带文件的文件的研究（Springer，2007），第231-250页

8.      ZM Shafiq，SA Khayam，M Farooq，*检测入侵和恶意软件及漏洞评估（DIMVA）*。使用markov n-grams嵌入式恶意软件检测（Springer，2008），第88-107页9.              文档管理 - 便携式文档格式 - 第1部分：PDF 1.7（2008）。[https://www.adobe.com/devnet/pdf/pdf\_reference.html。](https://www.adobe.com/devnet/pdf/pdf_reference.html)2015年1月23日访问

10.   P Laskov，NŠrndic在“ *年度计算机安全应用大会”上的发言*

*（ACSAC）*。静态检测恶意的带有JavaScript的PDF文档

（ACM，2011），第373-382页

11.   D Maiorca，G Giacinto，I Corona，*在模式识别中的机器学习和数据挖掘国际研讨会上*。用于恶意PDF文件检测的模式识别系统（Springer，2012），第510-524页

12.   C Smutz，A Stavrou，*第28届计算机安全应用会议论文集*。使用元数据和结构特征进行恶意PDF检测（ACM，2012），第239-248页

13.   NŠrndic，P Laskov *，在2014年IEEE安全和隐私专题讨论会论文集中*。基于学习的分类器的实际回避：案例研究

（IEEE计算机协会，2014），第197-211页

14.   M Polychronakis，KG Anagnostakis，EP Markatos，*年度计算机安全应用会议（ACSAC）*。使用运行时启发式技术的全面shellcode检测（ACM，2010），第287-296页

15.   P Akritidis，EP Markatos，M Polychronakis，KG Anagnostakis，*第20届国际信息安全会议*。STRIDE：通过指令序列分析进行多态滑板检测（Springer，2005），第375-392页

16.   Wepawet（2015）。[http://wepawet.iseclab.org/。](http://wepawet.iseclab.org/)2015年4月16日访问

17.   M Cova，C Kruegel，G Vigna在*万维网（WWW）国际会议上的发言*。检测和分析驱动器下载攻击和恶意JavaScript代码（ACM，2010），第281-290页

18.   C Willems，T Holz，F Freiling，CWSandbox：朝向自动动态二进制分析。IEEE Secur。隐私。**5**（2），32-39（2007）

19.   KZ雪，S克里希南，F Monrose，N Provos，在*USENIX安全研讨会*。ShellOS：实现代码注入攻击的快速检测和取证分析（USENIX Association，2011）

20.   Tang，S Sethumadhavan，SJ Stolfo *在攻击，侵入和防御研究：2014年9月17 - 19日在瑞典哥德堡召开的第十七届国际研讨会2014年RAID会议上的会议记录*。使用硬件功能进行无监督的基于异常的恶意软件检测，8688（Springer，2014），p。109

21.   Z Tzermias，G Sykiotakis，M Polychronakis，EP Markatos在*欧洲系统安全研讨会（EuroSec）上的发言*。结合静态和动态分析检测恶意文档（ACM，2011）

22.   X Lu，J Zhuge，R Wang，Y Cao，Y Chen，in *System Sciences（HICSS），2013 46th Hawaii International Conference On*。高精度地去混淆和检测恶意PDF文件（IEEE Computer Society，2013），第4890-4899页

23.   N Nissim，A Cohen，C Glezer，Y Elovici，检测恶意PDF文件和改进方向：一项最先进的调查。COMPUT。SECUR。**48**（0），246-266（2015）

24.   S Ford，M Cova，C Kruegel，G Vigna，*计算机安全应用会议，2009年ACSAC'09。年度*。分析和检测恶意Flash广告（IEEE计算机协会，2009年），第363-372页

25.   TV Overveldt，C Kruegel，G Vigna，*最近的入侵检测技术（RAID）*。FlashDetect：ActionScript 3恶意软件检测（Springer，2012），第274-293页

26.   NŠrndic，P Laskov *，于2013年2月24日至27日在美国加利福尼亚州圣迭戈*举办的*第20届年度网络与分布式系统安全研讨会，NDSS 2013*。基于分层文档结构检测恶意PDF文件（2013）。[http://internetsociety.org/doc/detection-malicious-pdf 基于文件的分层结构文档结构](http://internetsociety.org/doc/detection-malicious-pdf-files-based-hierarchical-document-structure)

27.   SWF文件格式规范（版本19）（2012）。[HTTPS：//www.adobe。COM / DEVNET / swf.html。](https://www.adobe.com/devnet/swf.html)2015年1月23日访问

28.   SWFRETools - 用于反向工程Flash文件的工具集合。[https://github.com/sporst/SWFREtools。](https://github.com/sporst/SWFREtools)2015年2月4日访问

29.   Hidost - 用于从PDF和SWF文件中提取文档结构的工具集。[https://github.com/srndic/hidost。](https://github.com/srndic/hidost)2015年11月29日访问

30.   波普勒。[http://poppler.freedesktop.org/。](http://poppler.freedesktop.org/)访问2015年2月10日

31.   ķ里克，T克鲁格，A Dewald，在*年度计算机安全应用大会（ACSAC）*。Cujo：有效的检测和预防驱动下载攻击，（2010），第31-39页

32.   ģWidmer分馏，男库巴特，在概念漂移和隐藏上下文的存在学习。马赫。学习。**23**（1），69-101（1996）

33.   Kantchelian，S Afroz，L Huang，AC Islam，B Miller，MC Tschantz，R Greenstadt，AD Joseph，J Tygar，*2013年ACM研讨会论文集*

*人工智能与安全*。对抗漂移的方法

（ACM，2013），第99-110页

34.   F Pedregosa，G Varoquaux，Gramfort，V Michel，B Thirion，O Grisel，M Blondel，P Prettenhofer，R Weiss，V Dubourg，J Vanderplas，A Passos，D Cournapeau，M Brucher，M Perrot，E Duchesnay， scikit-learn：Python中的机器学习。J.马赫。学习。RES。**12**，2825年至2830年（2011）

35.   布雷曼，随机森林。马赫。学习。**45**（1），5-32（2001）36.     Hidost复制。[https://github.com/srndic/hidost-reproduction。](https://github.com/srndic/hidost-reproduction)2015年11月29日访问

37.    VirusTotal - 免费在线病毒，恶意软件和URL扫描器。[https：//开头WWW。](https://www.virustotal.com/)

[virustotal.com/。](https://www.virustotal.com/)2015年3月6日访问