自动逃避分类器

PDF恶意软件分类器案例研究

徐维林，齐延军，*弗吉尼亚*大卫埃文斯*大学* http://www.EvadeML.org

*摘要 -*机器学习被广泛用于开发安全任务的分类器。但是，这些方法对于有动机的对手的稳健性是不确定的。在这项工作中，我们提出了一个通用的方法来评估受到攻击的分类器的鲁棒性。关键的想法是随机操纵一个恶意样本，以找到一个保留恶意行为的变体，但被分类器分类为良性。我们提出了一种通用方法来搜索回避变体，并使用我们针对两种PDF恶意软件分类器（**PDFrate和Hidost**）的技术进行实验。我们的方法能够为*所有的*分类器自动找到规避的变体在我们的研究中的500个恶意种子。我们的研究结果表明了评估安全应用中使用的分类器的一般方法，并且在存在对手的情况下，基于表面特征的分类器的有效性引起严重的怀疑。

                                      一        ，引言

机器学习模型在恶意软件检测，网络入侵检测和垃圾邮件检测等安全任务中非常流行。从数据科学家的角度来看，这些模型是有效的，因为它们在测试数据集上实现了极高的准确性。例如，达尔（Dahl）等人 报告使用具有动态特征的集合深度神经网络在对Win32恶意软件进行分类时达到了99.58％的准确性[9]。Srndic等人 在使用具有结构路径特征的SVM-RBF模型的PDF恶意软件分类任务中实现了超过99.9％的准确性[28]。

但是，重要的是要认识到这些结果是针对特定的测试数据集的。与在其他领域使用机器学习不同，安全任务涉及对手响应分类器。例如，攻击者可能会尝试生成新的恶意软件，故意设计来逃避现有的分类器。这打破了机器学习模型的假设：训练数据和操作数据共享相同的数据分布。因此，对安全环境中的机器学习结果持怀疑态度，而不考虑攻击者为逃避生成的模型所做的努力。

在机器学习领域已经讨论了在对抗设置下对机器学习模型的规避攻击的风险，主要集中在简单的垃圾邮件检测模型（例如[10,18]）。然而，在分类算法和特征提取以及高度结构化样本的可变性方面，针对恶意软件分类的规避攻击可能要复杂得多。

因此，尽管分类器作者和安全研究人员已经部分地探索了恶意软件分类器，但以前的研究显着低估了攻击者操纵样本的能力。例如，以前的研究可能错误地认为攻击者只能插入新的内容，因为删除现有的内容很容易破坏恶意[4,20,28]。此外，以前的作品是特设的，仅限于特定的目标分类器或特定类型的样本[20,29]。除了建议点解决方案之外，他们不提供自动评估分类器对抗自适应对手的有效性的方法。

我们提出了一种通用的方法来评估**分类器的鲁棒性**，通过模拟攻击者逃避分类器的努力。我们并不假定对手对分类器或其使用的特征有任何详细的了解，或者可以使用有针对性的专家知识来手动指引搜索避开样本。相反，从遗传编程（GP）[11,15]绘制想法，我们执行随机操作，然后评估生成的变种，以选择有前途的。通过迭代重复这个过程，我们的目标是产生回避变体。一个复杂的攻击者当然可以做一些不会被随机搜索找到的操作，所以我们不能说一个抵抗这种攻击的分类器必然是强大的。另一方面，如果自动方法找到给定分类器的回避样本，

我们在两个PDF恶意软件分类器上评估了所提出的方法，发现它可以自动找到从Contagio PDF恶意软件存档中选择的所有500个样本种子的回避变体[5]。回避变体展现出与原始样本相同的恶意行为，但是在分类器的特征空间中充分不同，被机器学习模型分类为良性。

我们对发现的回避变体的分析表明，这两个分类器都是易受攻击的，因为它们使用了非鲁棒的特性，可以在不中断所需的恶意行为的情况下进行操作。表面特征可能在测试数据集上运行良好，但是如果用于对恶意软件进行分类的特征是训练数据的浅层构件而不是恶意内容的固有属性，那么可以设法在破坏特征的同时保留恶意行为。贡献。我们的主要贡献涉及发展

允许自由复制本文件的全部或部分内容用于非商业目的，前提是复印件在第一页上注明本声明和全部引用。未经互联网协会事先书面同意，第一作者（仅复制整篇论文）和作者的雇主（如果该文件是在就业范围内编写的），严禁以商业目的复制。NDSS'16，2016年2月21 - 24日，美国加利福尼亚州圣迭戈版权所有2016 Internet Society，ISBN 1-891562-41-X http://dx.doi.org/10.14722/ndss.2016.23115

并评估一种自动寻找逃避分类器的变体的一般方法。尤其是：

•          我们提出了一种通用的方法来自动探索目标分类器的变体。该方法不依赖任何特定的分类算法或假设详细的特征提取知识，但只需要生成变体的分类评分反馈和分类器使用的可能特征的粗略知识（第II部分）。

•          我们实现了一个原型系统，可以自动查找可以逃避基于结构特征的变体

PDF恶意软件分类器。这包括设计对PDF文件进行随机操作的操作符，确定生成的变体是否保留恶意的预言器，在进化过程中促进有希望的变体的选择机制以及每个目标分类器的适应度函数（第IV节）。

•          我们评估了我们的系统在回避最近的PDF恶意软件分类中的有效性：PDFrate [25]和

Hidost [28]，一个分类器设计的明确目标是抵制逃避企图。在500个恶意软件样本种子的实验中，**我们的系统**可以100％的成功率找到两个分类器的回避变种。对每个分类器的特征空间中发现的回避变体的分析表明，分类中使用的许多非鲁棒特征有助于回避攻击（第V部分和第VI部分）。

我们在第二节提供机器学习分类器的背景知识，第三节提供有关PDF恶意软件的背景知识。第八节讨论了有关逃逸攻击的工作。

                                          II。       O VERVIEW

我们提出了一种自动的方法来模拟攻击者试图找到一个目标分类器检测到的恶意软件样本的回避变种。攻击者的目标是找到一个能够保留原始样本的恶意行为的恶意软件变体，但被目标分类器误认为是良性的。除了提高我们对分类器在自适应对手面前如何工作的理解，我们希望我们的结果将导致构建对对手更加强大的分类器的策略，但是在这项工作中，我们将重点放在评估evadability上。

*A.机器学习分类器*

机器学习从数据中学习并做出预测。基于机器学习的分类器试图找到将数据点映射到不同类的假设函数*f*。例如，恶意软件分类系统会找到一个假设函数*f*，将数据点（恶意软件样本）映射为*良性*或*恶意*。

训练机器学习系统的努力始于特征提取。由于大多数机器学习算法不能在高度结构化的数据上操作，所以数据样本通常在特别设计的特征空间中表示。例如，恶意软件分类器可以将文件大小和函数调用跟踪提取为特征。每个特征都是特征空间中的一个维度; 因此，每个样本都被表示为一个向量。当特征数量对于分类算法而言太大时，可以执行特征选择的额外步骤以减少特征的数量。

在安全任务中使用最广泛的机器学习算法使用*监督学习*，其中训练数据集带有标识每个训练样本的类别的标签。训练假设函数*f*以最小化训练集上的预测误差。在*平稳性*假设下，这个函数通常导致运行数据的低错误率，即将来遇到的数据点的分布将与训练集上的分布相同。

机器学习产生了令人印象深刻的结果，并广泛部署用于特定的安全任务，包括恶意软件分类 在不检查真实系统中可疑恶意软件的行为的情况下，恶意软件分类器通常会使用静态属性来预测恶意，如文件结构，文件大小，元数据，令牌克数或系统调用。尽管这种方法在验证测试中经常达到高精度，但是分类器可以学习属于训练数据的表面伪像的属性，而不是与恶意软件固有地相关联的属性。这是因为训练数据中的恶意软件样本可能与良性样本在许多方面有所不同，而这些样本对其恶意行为并不重要。

*B.威胁模型*

我们假设攻击者从一个被目标分类器（正确地）分类为恶意的期望的恶意样本开始，并且想要创建一个具有相同恶意行为的样本，但是被误分类为良性样本。攻击者能够以多种方式操作恶意样本，并且可能具有（正确地）分类为良性样本的知识。

我们假设攻击者可以对目标分类器进行黑箱访问，并且可以向该分类器提交多个变体。对于每个提交的变体，攻击者都会学习其分类分数。分类分数是一个数字（通常是介于0和1之间的实数），表示分类器对恶意的预测，其中高于某个阈值（比如0.5）的值被认为是恶意的，分类分数低的样本被认为是良性的。我们不假设攻击者有任何关于分类器的内部信息，只是它可以用它作为输入输入样本的分类得分的黑盒子。我们假设分类器操作符不会将分类器适配到提交的变体（如果攻击者已经离线访问分类器，则必须是这种情况）。

*C.找到回避样本*

我们的方法使用遗传编程技术来执行对可能样本的空间的定向搜索，以找到逃避分类器的样本，同时保持期望的恶意行为。

*遗传编程*（GP）是一种演化算法，最初被开发用于自动生成针对特定任务的计算机程序[11,15]。它本质上是一种随机搜索方法，使用生物变异和交叉的计算模拟生成变体，并使用用户定义的*适应度函数*对达尔文选择进行建模。选择适应度较高的变体进行持续进化，并且该过程继续进行多代，直到找到具有期望属性的变体（或者在超过资源边界之后终止搜索）。遗传编程在许多方面已被证明是有效的

|  |
| --- |
|  |
|  | E:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image006.gif |

图1.     通用分类器规避方法

包括修复遗留软件错误[17]，软件逆向工程[13]和软件重新设计[23]。

方法。我们的过程如图1所示。它从具有恶意行为的种子样本开始，并被目标分类器分类为恶意。我们的方法旨在找到一个避开样本，保留恶意行为，但被目标分类器错误分类为良性。

首先，我们通过对恶意种子执行随机操作来初始化变体种群。然后，每个变体都由一个目标分类器以及一个oracle进行评估。的*目标分类器*是一个黑盒子，其输出一个数字，是输入样本的预测的恶意的量度。有一个规定的门槛，用来决定是否恶意或良性。该*预言*被用于确定是否一个给定的样品显示出特定恶意行为。在大多数实例中，oracle将涉及昂贵的动态测试。

被目标分类器分类为良性，但被发现是恶意的变体是一个成功的回避样本。如果在群体中没有发现回避样本，则基于设计用于反映寻找回避样本的进度的适合度量来为下一代选择所生成的变体的子集。由于转换不可能将恶意行为重新引入到变体中，因此已经丢失了恶意行为的损坏变体被替换为其他变体或原始种子。

接下来，所选择的变体被变异算子随机操纵以产生下一代种群。这个过程一直持续到找到一个回避的样本或达到了一个世代的阈值。

为了提高搜索的效率，我们收集使用的变异操作的痕迹，并重用有效的痕迹。如果搜索结果发现任何回避变体，则回避变体上的变异痕迹将被存储为成功的痕迹。否则，存储具有最高适合度分数的变体的突变轨迹。这些跟踪然后被应用到其他恶意软件种子生成他们的人口初始化的变种。由于PDF的结构和突变算子的性质，相同的突变序列往往可以有效地应用于许多初始种子。

                   III。             PDF M ALWARE和C LASSIFIERS

本节提供有关PDF恶意软件和两个目标PDF恶意软件分类器的背景信息。

*A. PDF恶意软件*

该*可移植文档格式*（PDF）是一种流行的文档格式，旨在使在渲染和打印一致的内容和布局在不同的平台。尽管直到2008年才公开标准化[1]，并且有不同的PDF阅读器产品支持各种非标准扩展，所有PDF文件大致上具有图2所示的相同的基本结构。

一个PDF文件由四部分组成：*标题*，*正文*，*交叉引用表*（CRT）和*预告片*。该*标题*包含了PDF幻数和格式版本的指标。所述*主体*是一组包括所述文件的内容的PDF的对象，而在CRT的索引中的对象*体*。该*预告片*指定如何找到CRT等特殊对象，如根对象。因此，PDF阅读器通常从文件末尾开始阅读PDF以提高效率。

的*身体*是一个PDF最重要的，因为它拥有几乎所有可见的文档数据。它包含八种基本类型的对象，即布尔值，数字，字符串，名称，数组，字典，流和空对象。这些对象可以用一对整数标识符作为间接对象进行标记，以便它们可以被其他对象引用。相互引用的对象形成了一个树状的逻辑结构，如图2右侧所示。这个树状结构非常适合基因编程技术，因为它很容易改变和移动子树来生成新的变体。

PDF恶意软件正变得越来越普遍，因为PDF是一种被广泛接受的文档格式，受害者比其他文件更愿意打开PDF。根据最近的互联网安全威胁报告[30]，2014年，PDF在鱼叉钓鱼电子邮件中排名前7位。我们预计PDF恶意软件攻击将继续存在，因为Acrobat Reader中的128个新漏洞已经在CVE到2015年（到12月8日）为止，几乎是2014年的三倍[8]。

PDF恶意软件通常包含JavaScript对象或利用特定PDF阅读器（通常为Adobe Acrobat）的漏洞的其他对象的漏洞。

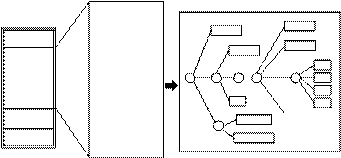


图2.       PDF文件的物理和逻辑结构

PDF恶意软件也可能在流对象中携带其他编码的有效载荷，这些对象将在利用后触发[25]。

*B.目标分类*

有几个项目使用机器学习技术构建了PDF恶意软件分类器。早期的作品，如Wepawet [7]和PJScan [16]，关注嵌入式恶意JavaScript中的PDF恶意软件。这些工具由JavaScript代码提取器和动态或静态恶意JavaScript分类器组成。

由于不是所有的PDF恶意软件都涉及嵌入式JavaScript，而且PDF恶意软件作者已经发现隐藏JavaScript代码的许多技巧[24]，所以最近的PDF恶意软件分类器已经集中在PDF文件的结构特征上。在这项工作中，我们针对最先进的基于结构特征的分类器。

基于结构特征的分类器假定恶意PDF在其内部对象结构中具有不同于良性PDF中的那些模式。例如，PDF恶意软件杀手工具使用对象关键字作为特征，其中每个特征对应于给定关键字的出现[19]。对于我们的实验，我们选择了PDFrate [25,26]和Hidost [28]作为目标分类器。他们是最近的PDF恶意软件分类器的代表，Hidost的开发具有回避攻击弹性的特定目标。这两个分类器在其测试数据集上都能够实现极高的恶意软件检测准确度。选择这些分类器作为我们的目标的另一个原因是开源实现的可用性。尽管我们的方法只需要黑箱访问分类器，

PDFrate。PDFrate是一个随机森林分类器，它使用一个集合学习模型，包含大量的旨在减少预测差异的决策树。利用训练数据的随机子集和特征的随机子集，对每个决策树进行训练以使其训练子集上的预测误差最小化。在训练之后，PDFrate的输出分数是输出“恶意”的树的分数，范围从0到1.阈值典型地为0.5，尽管PDFrate作者声称将阈值从0.2调整到0.8对准确度的影响不大因为大多数样本的得分非常接近0或1。

除了对象关键字之外，PDFrate还使用PDF元数据和对象的多个属性作为分类特征。PDF元数据包括作者，标题和创建日期。对象属性包括位置，计数和长度。

使用Contagio数据集的一个随机子集对PDFrate进行训练，使用5,000个良性和5,000个恶意PDF。这两个参数是分别的树木（数*ntree* = 1 *，* 000）以及特征在每个树的数目（*mtry* = 43）。该功能集共有202个整数，浮点和布尔功能，但PDFrate文档中只描述了其中的135个功能。

我们在这项工作中使用的是由Nedim Srndic和Pavel Laskov实施的模仿PDFrate进行恶意软件逃避实验的PDFrate命名为Mimicus [27]的开源重新实现[29]。Mimicus接受了135个有记录的PDFrate特征和与PDFrate相同的训练集。[[1]](file:///E:\\bdfirst\\2018\\2018Q1\\Automatically%20Evading%20Classifiers%20A%20Case%20Study%20on%20PDF%20Malware%20Classifiers.htm" \l "_ftn1" \o ") Mimicus已被证明具有与PDFrate [29]几乎相同的分类性能。

Hidost。Hidost是一个支持向量机（SVM）分类模型。支持向量机（SVM）是一种最佳的边缘分类器，它试图找到少量的支持向量（数据点），用高维空间的超平面来分隔两类的所有数据点。使用内核技巧，可以将其扩展为非线性分类器以适应更复杂的分类问题。Hidost使用径向基函数（RBF）核将数据点映射到无限维空间。在测试时间，数据点到超平面的（正或负）距离被输出作为预测结果。积极的距离被认为是恶意的，消极的是良性的。

Hidost使用对象的结构路径作为分类特征。例如，一个典型的Pages对象的结构路径是*/ Root / Pages*。如果该对象出现在PDF文件中，则其特征值为1; 如果不是，则其特征值为0.由于PDF对象的可能结构路径的数量是无限的，所以Hidost使用6087个选择的路径作为特征。所选择的路径是从VirusTotal [31]和Google搜索收集到的658,763个良性和恶意PDF文件池中的至少1000个文件中出现的路径。由Hidost的作者提供的模型是使用随机采样的5,000个恶意文件和5,000个良性文件进行训练的。据报道，对于对手而言，对另外5000个随机恶意文件的假阴性数量从28个增加到30个，据称这是“最强的可想象的模仿攻击”[28]。

               IV。              E VADING PDF M ALWARE C LASSIFIERS

所提出的方法可以应用于任何安全分类器，尽管其有效性取决于能够找到好的遗传程序设计算子来有效地搜索特征空间并且适当的适应度函数来指导搜索。在本节中，我们将展示如何实例化我们的设计，以找到避开PDF的恶意软件。

*A. PDF解析器和重新包装*

第一步是将PDF文件解析为树形表示。我们还需要从树形表示中重新生成一个PDF文件，在它被操作后产生一个新的变体。为此，我们使用pdfrw [21]，这是一个基于python的开源库，用于将PDF文件解析为树状结构，并将该结构序列化为输出PDF文件。

需要注意的是，pdfrw并不是一个完美的PDF解析器和重装器，而且大量的PDF恶意软件样本已经故意绕过或混淆了恶意软件检测器中使用的PDF解析器（同时由于解析器怪癖仍然被目标PDF阅读器处理）。这意味着我们无法在PDF种子样本上测试我们的方法，这些样本不能被pdfrw解析，或者在使用pdfrw进行解包和打包时不再显示恶意行为。

为了避免由于PDF解析问题而丢失太多样本，我们修改了pdfrw来放宽其语法检查。这显着提高了重新包装PDF恶意软件样本的成功率。pdfrw的修改版本可在https://github.com/mzweilin/pdfrw上找到。

在我们修改过的pdfrw中，我们忽略了几个可能被破坏的，畸形的或误导性的辅助元素。PDF原始字节中的*EOF*标记被忽略; 相反，解析器读取文件的所有字节。该*交叉引用表*被忽略; 相反，它直接解析*身体中的*对象而没有任何索引。流长度指标被忽略; 相反，解析器使用*endstream*检测流的长度令牌。解析字典时，未配对的键或值也会被忽略。忽略这些辅助元素会显着降低解析效率，因此仅适用于重新包装种子恶意软件样本。所有的种子都用正确的辅助元素重新打包，以便以后进行高效解析。另外，我们增加了对解析恶意软件样本中存在的空对象的支持。字典数据结构被修改以使得能够从种子复制变体的深拷贝。

*B.遗传算子*

由于我们定位的两个分类器都使用PDF文件的对象结构作为特征，因此我们需要通过在该级别上操作PDF文件来生成变体。（如果我们的目标是基于JavaScript的分类器，我们将需要通过操作嵌入的JavaScript代码来生成变体。）由于可能的静态特征的数量有限，我们认为假设攻击者知道操纵水平。

我们使用生物进化中的*突变*计算类似物来生成回避性的PDF恶意软件变体。变异算子以低概率改变PDF文件的树状结构中的任何对象。一个对象被变异概率给定的*突变率*，通常是一个小于0.5的数字。突变可以是*删除*（*删除*对象），*插入*（*插入*另一个对象后）或*替换*（此对象替换为其他对象）。

我们在这些选项中选择均匀的随机概率。在插入或替换的情况下，还从由良性PDF分割的大量对象中随机地均匀地选择第二对象。外部基因组有助于产生更多样化的人群。

另一个着名的算子，*交叉*，常用于遗传算法，没有在这项工作中使用。我们找到了

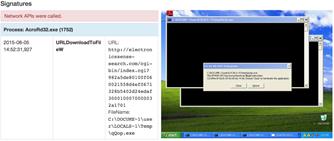


图3.由Cuckoo沙箱提供的PDF恶意软件检测结果。左侧显示关键的API执行轨迹，右侧是从虚拟机捕获的截图。

只有使用简单的变异操作才有可能达到100％的逃避率。

*C.甲骨文*

我们需要一个oracle来确定一个变种是否保留了种子的恶意行为。没有完全准确的恶意软件检测技术可以普遍使用（事实上，如果存在这样的技术，我们的工作就不需要了）。在这种情况下，我们有一个优势，可以使用高度准确的oracle来测试变体：我们不需要一个可以测试任意恶意行为的oracle，而只需要验证一个特定的已知恶意行为是由变体执行的。

要做到这一点，我们使用杜鹃沙箱[12]。Cuckoo在安装有PDF阅读器的虚拟机中运行提交的示例，并报告示例的行为，包括调用的网络API及其参数。图3显示了一个来自Cuckoo的恶意软件检测结果的例子。在虚拟机中打开的恶意软件样本利用了Acrobat Reader（CVE2007-5659）中发现的缓冲区溢出漏洞。注入的shellcode从互联网下载另外四个恶意软件并执行它们。由于Cuckoo的执行是从互联网上隔离的，为了避免恶意软件的传播，shellcode只是收到由网络服务模拟器INetSim提供的不正确的可执行文件[14]。但是，Cuckoo检测到的下载和执行行为足以表明shellcode已经被执行。通过比较原始PDF恶意软件的行为签名和操纵变体，我们确定原始恶意行为是否被保留。关于我们如何选择和比较行为签名的细节推迟到VA部分。

我们只关注恶意软件样本的网络行为。虽然此设置可防止我们的方法在没有网络活动的情况下处理恶意软件样本，但我们认为这并不是实际的限制，因为恶意软件作者总是可以开发一种方法来验证所需的恶意行为。

杜鹃沙箱可以很好的工作，但是计算量很大。我们尝试了其他可能的神谕，包括使用Wepawet。Wepawet和类似的检测技术只检测恶意有效载荷，但不能验证有效载荷是否在真正的PDF阅读器中被执行。因为许多基因突变会破坏执行，实际上没有动态观察显示恶意行为的变体的神谕导致许多误报（显然是不会实际上作为恶意软件的回避变体）。因此，使用一个确认通过实际执行保留恶意行为的oracle是非常重要的。这限制了我们可以在我们的实验中使用的样本，我们可以在我们的oracle的测试环境中产生恶意行为（第五节）。

*D.健身功能*

健身功能给出每个生成的变体的健身评分。分数越高越好。给定0作为阈值，具有积极健康分数的变体是回避的：它被分类为良性并且保留恶意行为。

在我们的例子中，适应函数捕获了oracle的输出和目标分类器的预测结果。oracle被建模为一个二进制函数：如果*x*表示恶意签名，则*oracle*（*x*）= 1 ; 否则，*oracle*（*x*）= 0。为了消除损坏的变体，我们总是给*oracle*（*x*）= 0的变量赋值最低的可能性。

基于目标分类器使用的不同评分方法，分别定义适应度函数。PDFrate作为随机森林分类器，输出从0到1的恶意的置信度值，通常阈值为0.5。因此，我们将它的适应度函数定义为

*健身pdfrate*（*x*）=⇢ *低* 0 *。*5 *pdfrateSCORE* （*x*） *oracleoracle*（（*xx*）= ）= 10

E:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image010.gif

用的回避范围（0 *，* 0 *。* 5 ] 。

Hidost的SVM模型输出一个良性（恶意）样本到超平面的负（正）距离。因此，对于Hidost，适应度函数被定义为

*健身hidost*（*X*）=⇢ *LOWhidost*（*SCOREx*）⇥（1 ） *oracleoracle*（（*XX*）=）= 10

E:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image010.gif

与回避范围（0 *，* + • ）。

*E.选择*

GP中的选择过程可以像在一代中一直选择具有更高健身评分的变体一样简单。然而，在进化过程中可能会发生一代甚至几乎没有变种保留恶意行为。如果恶意行为从人群中流失，那么GP就不可能找到显示原始恶意行为的回避样本。

为了避免人口退化，我们设计了一个替代机制，除了naïve的选择过程。被认为是非恶意的已损坏的变体被指定为最低的健身评分（*LOW SCORE*），并被原始的恶意PDF，迄今为止发现的最好的变体或者在先前发现的最佳变体代。我们在这些选项中选择了一致的随机概率，当发生损坏的变体时，这确保了在每一代中保留固定数量的变体。E:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image010.gif

*F.跟踪收集和重放*

初始化群体的最常见方法是复制原始种子，并在每个副本上执行随机变异操作。考虑到各种回避变体的潜在共同特性，我们通过重复使用成功导致回避或有希望的变体的突变痕迹来加速搜索。

一个突变轨迹由3元组（*变异操作符*，*目标对象路径*，*文件id：源对象路径*）定义的一系列突变组成。例如，

（插入，*/根/页/儿童/ 1*， *1：/根/页/儿童/ 4*）

将良性文件*1中*的外部页面对象插入到目标PDF文件中。第IV-B节定义了三种可能的变异算子。尽管*目标对象路径*与*源对象路径*具有相同的格式，但它们是不同PDF文件中的路径。的*目标对象的路径*是指该变体的对象，而*源对象路径*指向与所述指定的文件ID的外部良性文件的对象。

在每个GP搜索结束时，突变轨迹被添加到两个池中。如果GP搜索成功生成回避变体，则所有相应的变异跟踪都会添加到*成功跟踪池中*。否则，将产生具有最高适应性分数的变体的突变轨迹添加到*有希望的轨迹库中*。

两个池中的踪迹在群体初始化中重播，以产生第一代的一些变体。如果可用的痕迹数量小于种群大小，则以常规方式生成其他变体。如果数量大于种群规模，则第IV-E节中所述的选择过程将使种群缩小到指定的大小。

                                        V.         E XPERIMENT

我们通过对两个目标PDF恶意软件分类器进行实验来评估所提出方法的有效性。

*A.数据集和实验设置*

我们从Contagio档案中的10,980个PDF恶意软件样本开始[5]，从中选择了500个合适的样本进行评估。这些样本被oracle验证为表现恶意行为，被两个目标分类器分类为恶意的，并且可以被pdfrw正确地重新打包。

恶意PDF数据集。表一总结了样本选择程序。

                                 表一           小号EED选择。

|  |  |
| --- | --- |
| 描述 | 数 |
| PDF在Contagio的恶意软件样本 | 10,980 |
| Wepawet检测到具有网络API调用的示例 | 9,688 |
| 杜鹃观察到网络活动的样本 | 1414 |
| 由pdfrw正确重新包装的独特样本 | 1384 |
| PDFrate的真正积极 | 1378 |
| Hidost真正的积极 | 502 |
| TP在PDFrate和Hidost中的交点 | 500 |

首先，我们通过Wepawet的shell代码分析筛选出没有任何网络API调用的样本，在10,980个样本中留下9,688个样本。这对于我们的方法来说不是必要的，但是很有用，因为我们使用Wepawet来获得关于样本的附加信息。

其次，剩下的样品在杜鹃沙箱中进行测试。根据Wepawet提供的每个样本的漏洞信息，除了针对福昕阅读器的CVE-20099837之外，Adobe Acrobat Reader 8.1.1是最常见的目标PDF阅读器。因此，这些样本装载了Acrobat Reader 8.1.1。然而，由于虚拟机中的网络仿真不完善，以及恶意软件潜在的沙箱检测特性，我们选择了一个有针对性的PDF阅读器，并不是所有的网络行为都可以在Cuckoo中被观察到。结果，在9,688个样本中只有1414个被发现在Cuckoo沙箱内的Acrobat Reader 8.1.1上运行恶意网络活动。

接下来，将1414个样本用修改后的pdfrw重新打包，语法检查不太严格，然后由Wepawet和Cuckoo重新测试。这导致了1384个独特的样本。十一个样品在重新包装过程中被破坏，不再在Wepawet或Cuckoo中恶意行为。其他十九个样本被重新包装后发现是重复的。这是一个明显的迹象，表明恶意软件作者试图通过解析混淆来逃避检测。

由于我们的目标是评估回避攻击的有效性，我们需要滤除目标分类器的假阴性样本。PDFrate正确地将1384个样本中的1378个分类为恶意，而Hidost只将502个正确分类。两个分类器的真正好处的交集留下了500个独特的PDF恶意软件样本的合适评估集。

根据Wepawet的结果，这500个恶意软件样本在Acrobat Readers中利用了两个不同的漏洞：其中333个利用CVE-2007-5659中报告的多个缓冲区溢出，另外167个利用CVE-2009-0927报告的基于堆栈的缓冲区溢出。这两个漏洞都可以被利用来执行任意代码。总之，500个样本中的有效载荷访问255个不同的主机，从互联网上下载额外的恶意软件。

在选择过程中，我们从Contagio档案库中最初的10,980个恶意软件样本中抽取了500个样本。尽管这个选择的样本不到原始样本的5％，但只要选择标准没有影响我们的结果的偏见，就不会影响到恶意软件作者试图找到回避样本的成功率。许多down-selection是由于实验的人为因素，不反映真正的恶意软件制造商会观察什么。例如，最显着的减少是因为我们选择用来验证恶意行为的特定动态环境。恶意软件作者可以轻松设计一个Oracle，验证他们打算造成的特定恶意行为的存在。

可靠的恶意软件签名。由于恶意软件样本的动态行为可能因执行而有所不同，因此我们需要从一组候选人中选择可靠的恶意软件签名。即使恶意软件在同一虚拟环境中执行，其行为也可能受到事件时间，服务故障以及其他非确定性来源的影响。

着眼于恶意软件样本的网络行为，我们可以提取由Cuckoo报告的各种网络行为作为签名，如DNS查询，HTTP URL请求和网络目的地。杜鹃从网络相关的API执行跟踪和捕获的网络流量生成这些报告。表二比较了从杜鹃报告中提取的六种不同类型签名的有效性。

我们在杜鹃虚拟机上测试了500个恶意软件种子，每个种子运行十次。我们的目标是确定哪种类型的签名在捕获观察到的恶意行为方面具有最佳的精确度，同时在同一个样本的多次执行中保持一致。

如果一个签名在十个测试的任何一个中提取了一个种子的任何相关行为，我们就把这个签名计算在这个种子上。显然，一个理想的签名将是有效的所有500个种子。我们还测量了10次重复测试中签名的一致性。我们将在十个测试中观察到的提取行为指定为种子的参考签名。种子的一致性按照*模式*10进行计算（即参考签名在10次试验中出现的次数的百分比）。

表II列出了每种种子在10次处理中每种签名的平均和最小一致性。通常，从API跟踪中提取的签名比从网络流量中提取的签名更加一致。我们选择从URL跟踪中提取的HTTP URL请求和主机查询的联合作为我们实验的签名。通过结合这两个行为签名，我们获得了对所有500个恶意软件种子有效的签名，并且具有最高的平均值和最小值一致性。

良性PDF数据集。我们使用带有filetype：pdf 和没有关键字的Google搜索收集了一组179个良性PDF文档。VirusTotal [31]和Wepawet [7]都确认所有文件都是良性的。我们只包含小于1MB的文件，以避免引入不必要的计算成本来处理极大的PDF文件。我们从目标分类器中选取3个良性最低（即最良性）的良性样本作为实验外部对象的来源。我们的结果显示，只有少数良性样本足以产生成功的逃避攻击。

GP参数。几个GP参数是任意选择的，没有任何参数微调，除了一个明显的限制：我们希望实验在合理的时间内完成。人口规模为48人，最大代数为20人，突变率为0.1。健身停止阈值为0.0，这表明已经找到了回避变体。

目标分类器。由于我们不想通过提交过多的自动生成的恶意软件变体来滥用在线部署的恶意软件分类系统，因此我们始终偏好本地可执行代码。我们使用了PDFrate和Hidost分类器的Mimicus重新实现，如第III-B部分所述配置和训练。

机。我们在实验中使用了一台典型的台式机（Intel Core i7-2600 CPU @ 3.40GHz和32GB物理机

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表二。                 ç OMPARISON网络的- 基于恶意软件特征。   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  | 一致性 | | | 资源 | 描述 | 例 | 有效 | 平均 | 最低限度 | | API跟踪 | HTTP URL请求和主机查询的组合 | [http://stortfordaircadets.org.uk/flash/exe.php?x=pdf，stortfordaircadets.org.uk] | 500 | 0.95 | 0.50 | | API跟踪 | 主机通过getaddrinfo（）查 询 | [stortfordaircadets.org.uk] | 497 | 0.95 | 0.50 | | 网络流量 | 传输层目标IP地址 | （udp：[192.168.57.2:53]，tcp：[192.168.57.2:80]） | 476 | 0.85 | 0.10 | | API跟踪 | 通过原始套接字请求的网址，  URLDownloadToFileW（），InternetOpenUrlA（） | [http://stortfordaircadets.org.uk/flash/exe.php?x=pdf] | 473 | 0.95 | 0.50 | | 网络流量 | DNS查询 | [stortfordaircadets.org.uk] | 462 | 0.93 | 0.10 | | 网络流量 | HTTP URL请求 | [http://stortfordaircadets.org.uk/flash/exe.php?x=pdf] | 460 | 0.93 | 0.10 | |

内存运行64位Ubuntu 14.04服务器）。Cuckoo沙箱由16个运行Windows XP SP3 32位和Adobe Acrobat Reader 8.1.1的虚拟机实例组成。使用我们的方法找到规避样本所需的资源是现成的。

                                           VI。       R 成果

基于GP的方法在回避两个目标分类器方面取得了惊人的好结果。对于这两个分类器，它能够生成一个保留恶意行为的变体，但是在我们的测试集中被分类为所有500个种子的良性。我们的代码和数据可以从http://www.evadeML.org 获得开源许可证

*A. PDFrate*

在执行大约一周后，该算法发现了72个有效的变异轨迹，为500个恶意软件种子产生了总共16,985个回避变体（平均每个种子34.0个回避变体），在攻击PDFrate时实现了100％的回避率。

痕量分析。所有产生回避变异的突变痕迹在之后的500个种子中重新执行，以研究每个痕迹的效力。这里的功效是测量应用给定踪迹的恶意软件种子的数量会产生一个回避的变体。

每个突变轨迹的长度及其功效如图4所示。轨迹按迹线ID进行排序，这反映了迹线的发现顺序。从图中我们可以看到，随着进化的进行，该方法通常会发现更长的变异痕迹。部分原因是后来种子的初始种群是使用收集的痕迹生成的。如果这些最初的变种不是回避，随后的突变将被添加到原始痕迹。

每个种子的功效与其长度没有强相关性。一个由插入页面对象的单个操作组成的变化轨迹为155个恶意软件种子生成了回避变体。还有189个操作的突变痕迹，只有两个种子有效。

图5给出了按突变长度排序的累积逃逸（为了比较，图中还显示了Hidost的结果，我们将在后面讨论）。生成变种以逃避PDFrate的难度在种子上大不相同。它只需要15个短的突变痕迹（不超过45个操作）为500个种子中的400个产生回避变体。为其他100个种子寻找回避变种，长度为48到354，长度为57个。

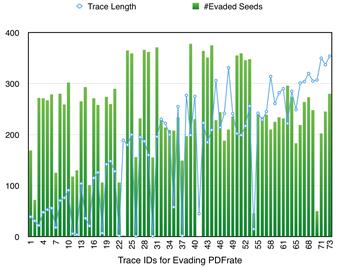


图4.      逃避PDFrate的突变痕迹的长度和功效。

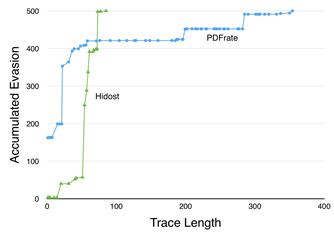


图5.对PDFrate和Hidost的累计回避，按照走线长度排序。

为了理解为什么需要更长的时间来为这100个种子产生回避变体，我们检查了每个种子的原始分类分数。图6通过生成规避变体所需的最小迹线长度对种子进行分组。左侧显示PDFrate中的原始分类分数分布。我们发现，分类得分较低（*<* 0.95）的原始种子大部分是通过短的痕迹来检测的。因此，我们相信一些种子需要更多的突变来逃避，因为它们本来对分类器更加明显地是恶意的。（在我们稍后讨论时，这在Hidost中更为明显）。

特征分析。为了理解躲避攻击，我们研究了这些变化对PDFrate使用的特征空间的影响。

我们首先看看对162个种子有效的两个最简单的长度为1的突变痕迹：

（插入，*/根/页/童装*，

*3：/ Root / Pages / Kids / 4 / Kids / 5 /*）

（替换，*/ Root / Type*，*3：/ Root / Pages / Kids / 1 / Kids / 3*）

即使它们是不同的操作，两种突变的共同作用是它们都从外部良性PDF引入新的页面对象，导致PDFrate的特征空间发生显着变化。

表III列出了通过简单插入几个页面对象的功能更改的一个例子。原始种子分类得分为0.998，接近最大恶意得分1.0。插入新的Page对象后，分类分数降低到0.43，低于正常的恶意软件阈值0.5。简单的插入导致了特征空间的大量变化。由于新引入的对象，某些对象（如页面，字体和流）的计数器以及文件大小会直接增加。对象长度统计量由于对象种群的变化而减少或增加。对象位置上的其他一些特征也由于在原始字节级重新定位对象而改变。所有特征值都是原始格式，因为随机森林不需要特征标准化。即使功能变化非常显着以至于PDFrate会将新变体分类为良性，但原始种子的恶意行为根本不会改变。更改只是将一些页面添加到PDF文件。

一个简单的操作引入了许多特征变化，但由于特征在分类中的重要性不同，所以改变每个特征的影响并不相同。虽然随机森林是一个难以解释的复杂的非线性模型，但是我们估计独立地改变每个特征的影响。直观地说，改变高影响力特征应该显着影响分类评分。

对于原始恶意软件样本，如果特征在回避变体中更改为不同值，则分类分数应该降低（表明它更像分类器中的良性样本）。另一方面，对于一个规避的变体，如果特征值被重置为原始值，则分类分数应该增加（对于分类器而言，更像是恶意样本）。因此，我们用两个因素对影响进行建模。当特征被改变为回避值时，恶意软件种子的分类得分的降低被报告为D *得分* 1.当特征被重置为原始值时，回避变体的分类得分的增加被报告为D *得分*2.影响是两个分数的总和。表3列出了受影响的特征的影响等级，其与PDFrate [26]中的特征重要性等级大致相符。

这个例子中最重要的功能变化是*count 字体*。原始恶意软件样本没有任何字体对象，因为漏洞不需要字体。分类器知道这个特征是重要的，因为大多数E:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image010.gif

                     表三。             我MPACT的PDF 率˚F 功能设定。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 特征 | 原版的 | 逃避的 | Dscore1 | Dscore2 | 碰撞 |
| 计数字体 | 0.0 | 70.0 | 0.114 | 0.392 | 0.506 |
| 计数obj | 11.0 | 230.0 | 0.067 | 0.110 | 0.177 |
| 计数endobj | 11.0 | 230.0 | 0.056 | 0.069 | 0.125 |
| 计数箱等 | 3.0 | 140.0 | 0.038 | 0.043 | 0.081 |
| 计数终端 | 4 | 74.0 | 0.011 | 0.054 | 0.065 |
| 邮箱最多 | 0.0 | 0.8 | 0.052 | 0.013 | 0.065 |
| 计数流 | 4 | 74.0 | 0.021 | 0.041 | 0.062 |
| 邮箱平均 | 0.0 | 0.5 | 0.022 | 0.022 | 0.044 |
| 平均值 | 1.0 | 1.0 | 0.000 | 0.032 | 0.032 |
| 最后一分钟 | 1.0 | 1.0 | -0.002 | 0.029 | 0.027 |
| 最多pos页 | 0.0 | 0.8 | 0.003 | 0.018 | 0.021 |
| 最大值 | 1.0 | 1.0 | 0.002 | 0.016 | 0.018 |
| pos页面平均 | 0.0 | 0.5 | 0.002 | 0.010 | 0.012 |
| 尺寸 | 36,028.0 | 503,739.0 | -0.001 | 0.005 | 0.004 |
| 比率大小页面 | 36,028.0 | 7,407.9 | 0.001 | 0.002 | 0.003 |
| 比例大小obj | 3,275.3 | 2,190.2 | 0.000 | 0.002 | 0.002 |
| 比例大小的流 | 9,007.0 | 6,807.3 | 0.002 | 0.000 | 0.002 |
| len obj avg | 3,234.9 | 2,157.2 | 0.001 | 0.000 | 0.001 |
| 计数页面 | 1.0 | 68.0 | -0.004 | 0.004 | 0.000 |
| len obj max | 27,455.0 | 34,314.0 | 0.000 | 0.000 | 0.000 |
| len obj分钟 | 44.0 | 49.0 | 0.000 | 0.000 | 0.000 |
| len流平均 | 8,700.3 | 6,390.2 | 0.000 | 0.000 | 0.000 |
| len流最大 | 27,392.0 | 34,246.0 | 0.000 | 0.000 | 0.000 |
| 最小页面 | 0.0 | 0.0 | -0.002 | 0.000 | -0.002 |
| 邮箱分钟 | 0.0 | 0.0 | -0.003 | 0.000 | -0.003 |

选择的种子样本（得分为0.998）和相应的回避变体（得分为0.43）的PDFrate的特征空间的差异。原始是原始种子恶意软件样本的特征值; 回避是回避变体的特征价值。不包括两个样品具有相同值的特征。D *得分* 1是原来的分数减去分数后，该特征改变为Evasive值; D *分数* 2是在特征重置为原始值之后减去*分数*的回避分数。影响是D *分* 1 + D *分* 2。

的恶意软件样本不包含任何字体对象，因为恶意软件作者懒得插入任何文本，但是不太可能任何良性PDF文件都没有字体对象。但是，这是训练集中恶意软件样本的人为因素，而不是恶意PDF的固有属性。将字体对象添加到现有的PDF恶意软件样本以改变此功能的值是微不足道的。

有更长的痕迹最多包含354个突变，影响PDFrate中的更多特征。表4列出了发现的所有16,985个回避变体中经常增加和减少的特征。（在附录A中找到了在逃避攻击中找到的PDFrate的所有68个可变特征的完整列表）。计数是发现的特征值与原始种子相比的多少倍。高计数意味着这些功能不健壮，不应该用于恶意软件分类，因为它们很容易改变，而不会破坏许多恶意软件种子的恶意属性。

由于PDF恶意软件作者可以随时更改PDF恶意软件样本中的可见内容（例如页面，文本，图像和元数据），而不会破坏恶意负载，因此大多数非强健功能并不令人惊讶。唯一令人惊讶的功能是*计数JavaScript*。由于PDF恶意软件严重依赖于JavaScript来携带漏洞利用代码和shell代码，因此可以在不中断恶意行为的情况下减少*javascript数量*，这似乎令人惊讶。但是，*计数的JavaScript*E:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image010.gif功能并不是PDF中嵌入的JavaScript代码段数量的准确计数。它只是提取JavaScript关键字的数量，但这些关键字在脚本执行中是可选的。有针对性的PDF阅读器即使没有*/ Javascript*关键字也会执行JavaScript 。

|  |
| --- |
|  |
|  | E:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image016.jpg |

图6.     种子原始分类评分的分布。

          表IV。                中号OST 一个过滤的˚F 功能设定ě VADING PDF RATE

|  |  |  |  |
| --- | --- | --- | --- |
| 计数 | 增加的功能 | 计数 | 功能下降 |
| 16985 | 平均值 | 14234 | 最小页面 |
| 16985 | 最大值 | 10,806 | len obj分钟 |
| 16985 | 最后一分钟 | 10728 | 计数的JavaScript |
| 16985 | 尺寸 | 8834 | len流分钟 |
| 16975 | 计数终端 | 7637 | 比例大小的流 |
| 16975 | 计数流 | 4,742 | 创造tz |
| 16,941 | 计数endobj | 4,742 | 三角洲tz |
| 16,941 | 计数obj | 4,250 | 比率大小页面 |
| 16862 | len流最大 | 3,448 | len流平均 |
| 16812 | 邮箱最多 | 3,137 | pos页面平均 |

*B. Hidost*

逃避Hidost的实验花了两天左右的时间来执行。虽然Hidost是专门设计成抵抗逃避尝试，[[2]](file:///E:\\bdfirst\\2018\\2018Q1\\Automatically%20Evading%20Classifiers%20A%20Case%20Study%20on%20PDF%20Malware%20Classifiers.htm" \l "_ftn2" \o ")我们的方法实现了100％的回避率，总共为500个种子（每种子5.7回避样品中平均）产生2859个回避样品。

痕量分析。我们分析每个突变谱的功效，这与用于PDFrate的相同方式一样。每个突变轨迹的长度和功效如图7所示。一般来说，它需要较短的突变痕迹，以达到100％攻击Hidost的回避率，而不是PDFrate。

我们观察到与PDFrate相比有两个主要区别。首先，新发现的变异曲线没有增加迹线长度的趋势，这与PDFrate中迹线长度随着迹线ID增加而不同。其次，迹线长度与效果更相关：较长的痕迹往往更有效地产生回避变体。少于5个突变的几个短突变痕迹仅对1或1有效

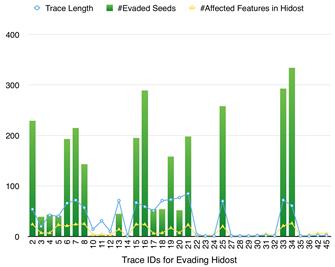


图7.     逃避Hidost的突变痕迹的长度和功效。

2个恶意软件种子。相比之下，包含61个突变的长突变痕迹在334种恶意软件种子上有效。

图5给出了根据突变长度进行排序的累计逃逸数量。该图更接近线性，这表明与PDFrate相比，寻找不同种子的避让变体的难度几乎没有变化。我们相信与PDFrate的不同之处在于Hidost中设置的不同功能。变异操作对Hidost中的结构路径特征有更直接的影响。例如，对象删除操作只是删除一个特征的相应路径（以及它的后代的路径）。相比之下，由相同的操作导致的PDFrate的特征变化较不明显。除了减少我们可以预期的特定对象的数量外，其他位置特征也可能由于重新包装修改的变体中的对象的重新定位而改变。因此，逃避PDFrate的突变操作之间存在更多的相互影响，并且可能需要大量的突变来达到逃避阈值。在图6右侧所示的每个种子的Hidost中的原始分类得分的箱形图表明，通常需要更多的突变来找到对分类器看起来更明显是恶意的种子的回避变体。

特征分析。Hidost中使用的二进制特征比PDFrate使用的各种特征更易于解释。

我们首先看最简单的突变痕迹。有5个长度为1的变异痕迹，仅对1或2个恶意软件种子有效。他们是：

（删除，*/ Root / OpenAction / JS /长度*）

（删除，*/根/名称*）

（删除，*/根/ AcroForm / DR*）

（替换，*/根/ AcroForm / DR*，

*3：/Root/OpenAction/D/0/.../FontBBox/3*）（取代，*/根/ AcroForm / DR*，

*3：/Root/Pages/Kids/3/.../DescendantFonts/0/DW*）

前三个突变每个从原始恶意软件种子中删除节点，将相应的Hidost功能的值从1更改为0.第一个删除的对象类似于PDFrate中的*计数javascript*功能。捕获恶意软件样本中经常存在但不是良性文件中的属性。但是，它们在恶意代码执行中是可选的。其他删除的对象是训练数据集中的与恶意行为没有紧密联系的工件。尽管最后两条曲线使用替换操作，但替换的重要作用是删除从原始*/ Root / AcroForm / DR*节点的子对象中提取的特征。E:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image010.gif

简单地删除一些对象不足以逃避Hidost（它只对我们实验中的1或2个恶意软件种子有效），但是额外的突变足以为所有种子找到回避变种。最长的变异轨迹包含85个操作，这对于198个恶意软件种子生成绕过Hidost的规避变体是有效的。表V列出了执行该变异跟踪时，在198个恶意软件种子上观察到的所有功能变化。不出所料，添加或删除了几个辅助对象来欺骗Hidost。例如，插入几个元数据对象。当用户使用流行的PDF编写器生成PDF文档时，元数据广泛存在于良性PDF中。另一方面，由于恶意软件作者没有添加手工制作PDF漏洞的元数据，所以在PDF恶意软件中很少见。然而，这只是训练数据集中的一个工件，并不是PDF文档和PDF恶意软件之间的根本区别。将元数据插入PDF恶意软件样本会增加Hidost对样本进行良性考虑的可能性。

从这个例子可以看出，走线长度本身并不是一个很好的逃避复杂度。尽管随机搜索过程发现了85个操作轨迹来创建这些规避变体，但轨迹仅影响表5中显示的23个特征（每个对应于PDF文件中的一个节点）。也就是说，操作轨迹也是一样有效的（可能更短的轨迹，因为一个突变可能影响许多特征），搜索找到的轨迹包括许多无用的或多余的突变。为了创建回避恶意软件，找到最短的有效跟踪并不重要，尽管这是可能的

表五      ˚F EATURE变化所生产LONGEST ^ h IDOST突变TRACE 。

|  |  |
| --- | --- |
| 新增功能 | 已删除的功能 |
| 主题 | AcroForm |
| ViewerPreferences /方向 | 名称/ JavaScript的/姓名/ S |
| 元数据 | AcroForm / DR /编码/ PDFDocEncoding的 |
| 元/长度 | AcroForm /.../ PDFDocEncoding的/差异 |
| 元/亚型 | AcroForm /.../ PDFDocEncoding的/类型 |
| 元/类型 | 页/旋转 |
| OpenAction /目录 | AcroForm /场 |
| OpenAction /内容/过滤器 | AcroForm / DA |
| OpenAction /内容/长度 | 大纲/类型 |
| 页/媒体框 | 纲要  概括/计数  页面/资源/ ProcSet页面/资源 |

开发技术，如果需要自动削减痕迹的基本操作。在黄色的三角形情节

图7显示了每个跟踪的受影响功能的数量。

尽管其作者声称Hidost对于涉及功能添加的回避攻击是强大的，但是我们发现了许多仅仅增加了功能的回避变体。在2,859个回避变种中，761个是纯特征加入攻击，其中21个是纯特征删除攻击，其余2077个涉及特征添加和删除。假设攻击者只能插入特征已经是不现实的，并且如关于Hidost的不可避免性的权利要求所示，由于特定的手动技术失败，所以认为技术不可避免是危险的。

附录B给出了避开Hidost中的变异特征的完整列表。这些非鲁棒特征不应该用在恶意软件分类器中，因为它们可以在保留原始恶意特性的同时进行轻松更改。

*C.交叉逃避效果*

即使分类器设计得非常不同，并且使用不同的训练数据集进行训练，但我们怀疑他们必须在相同的分类任务中共享一些属性。因此，我们进行了一个交叉逃避实验，通过给一个分类器提供回避另一个分类器的回避变体。

对于388种恶意软件种子，通过躲避Hidost发现的逃避变种也可以有效避免PDFrate。也就是说，在没有任何访问PDF的情况下，访问Hidost的恶意软件作者可以为77.6％的种子找到回避变种。相比之下，通过回避PDFrate找到的回避变体只对两个恶意软件种子的Hidost有效。

交叉躲避效应的显着差异是由于两个分类器中的不同特征集所致。事实上，Hidost的主要设计目标是通过使用基于结构特性的特征，而不像其他分类器那样容易被回避。该算法避开PDFrate产生的避让变量显着改变了测量特征，但对Hidost中使用的结构特征影响不大。在相反的方向上，通过直接改变结构特征（逃避Hidost所必需的）来针对Hidost的回避变体偶然影响到PDFrate所使用的特征。

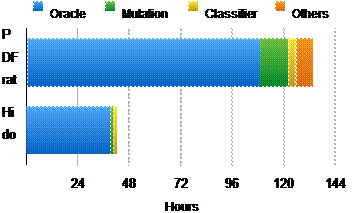


图8.     找到500个恶意软件样本的规避变体所需的时间。

*D.执行成本*

演化算法的一个缺点是它们不能保证在特定的时间内产生好的结果。对于许多问题，这些方法可能需要大量的计算资源才能找到想要的结果。而且，找不到理想的结果可能是它不存在的标志，或者只是需要更多的计算资源。我们的实验表明，这种实例化所需的资源是非常合理的。

对于每个分类目标，实验分几轮进行。第一轮从空轨迹池开始，所以搜索规避的变体完全依赖于由适应度函数指导的随机搜索。在以后的几轮中，以前发现的成功和有希望的突变痕迹被用来加速搜索。所有在一轮中失败的工作都会在下一轮中重新运行，并存储在前一轮中的所有变化轨迹。

对于PDFrate来说，花费了四轮才能达到100％的回避率。Hidost的回避率在第二轮达到100％。图8显示了逃避这两个分类器所需的全部500个恶意软件种子的回避变体所需的总时间。每个工作的持续时间没有意义，因为它主要取决于工作顺序。后来的工作通常要快得多，因为他们可能从之前发现的变异痕迹中受益，但尝试种子的顺序是任意的。

花了不到6天时间才达到PDFrate的100％回避率。换句话说，我们的方法在16分钟内平均在商品桌面上运行，发现了每个种子的回避变体。被发现的回避变种

Hidost快三倍，平均每个种子需要5分钟。

主要的计算时间是在Cuckoo沙盒中运行生成的变体，我们在这个过程中使用它作为oracle。具有16个并行运行的虚拟机的机器能够每小时测试1000个样本。这可以通过使用更多的机器来加速，因为执行之间没有依赖关系。

我们还观察到，攻击PDFrate的其他任务（包括变种）所花费的时间占总持续时间的比例比Hidost（8.3％比4.1％）要大。这是因为用作外部对象基因组的良性文件大于攻击Hidost的文件。因此，它产生了更大的变体，增加了解析，操纵和重新包装的计算负担。

                                       七。        D 讨论

在本节中，我们将讨论我们的结果提出的潜在防御和未来方向。

*A.防守*

除了理解当前分类器的漏洞，我们的最终目标是提高分类器受到攻击的鲁棒性。基于我们生成的规避样本以及我们在第五节中发现的非鲁棒特征，我们考虑了几种可能的方法。

信息隐藏和随机化。保护分类器的最直接的解决方案之一是隐藏用户的分类分数或者将随机噪声添加到分数[2]。另一个提出的方法是多分类器系统，其中分类分数是从不同的模型训练有些随机挑选不相交的特征[3]。由于我们的方法在很大程度上依赖于变体的分类分数来计算指导进化的适应性分数，所以缺少准确的分数反馈使得对回避性变体的搜索变得困难得多，并且可能使我们的方法不可行。

但是，表面特征的内在非鲁棒性不应该被忽略。考虑到潜在的交叉逃避效应（VI-C部分），隐藏或随机化信息可能无助于对付能够推断目标分类器使用的特征类型的对手。此外，以前的工作表明，准确地重新实施一个与替代训练集相似的分类器是可能的（实际上，这是Mimicus的作者做的实验PDFrate [26,29]的evadability）。

适应回避变体。我们的实验假设对手可以测试样本而不暴露给分类器操作员。在线情况下，分类器可以适应已尝试的变体。但是，请注意，再培训费用昂贵，打开了分类器以交替避税策略，如中毒攻击。

Chinavle等人 提出了一种方法，可以自动地重新训练带有伪标签的分类器，一旦通过集合模型上的相互协商措施检测到了回避变体，这已经在垃圾邮件检测任务中显示出有效[6]。然而，适应用户的输入而没有真正的标签会引发中毒攻击的新风险。

打败过度。我们演示的分类器的可避免性可能只是一个过度拟合的问题，在这种情况下，众所周知的机器学习实践应该能够克服过度拟合。例如，收集更大的数据集来训练模型，或使用模型平均来降低方差。

然而，我们并不期望这些传统方法会有所帮助。收集未来恶意软件的完整数据集是不可能的，而且这些技术都不能预测到积极企图逃避分类器的对手。

选择强大的功能。在回避实验中，我们发现了两个分类器的许多非鲁棒特征。

显然，它们应该从功能集中删除，因为攻击者可以轻松操纵它们而不会破坏恶意属性。然而，Hidost和PDFrate所使用的特征的问题在于，*所有*的特征都可能不健壮。这些分类器所使用的表面特征在良性和恶意PDF之间没有任何内在的区分性，如果发现可用于鲁棒分类的表面特征，将是非常令人惊讶的。相反，似乎有必要使用更深的特征来建立分类器，以抵御复杂的对手的逃避企图。这些功能将取决于对输入文件的更高级的语义分析，难以改变的方式在不破坏恶意行为的情况下进行。

*B.提高自动逃避*

我们的自动逃避方法提供了评估安全任务的分类器的鲁棒性的一般方法。它能够找到针对目标分类器的回避变体，显示出明显的弱点，但是如果我们的方法无法找到针对特定分类器的回避变体，那么肯定不足以确信其他技术（包括手工努力）将不能够找到回避变种。因此，改进该方法以使更有效的搜索能够针对更具挑战性的分类器是有价值的。

参数调整。在这项工作中，我们只是任意选择搜索参数。调整参数，甚至尝试像参数衰减这样的动态机制可以使搜索算法更高效。

可学习的GP。我们用来生成回避变体的当前方法基本上是一个随机搜索算法。因此，它通常会生成损坏的变体，从而失去恶意行为。一个概率模型可以了解哪些突变更有效地产生回避变体来更有效地指导搜索。

其他应用。我们的案例研究主要集中在PDF恶意软件，但是我们相信类似的方法可能对其他基于机器学习的恶意软件分类器有效。将我们的方法应用于新领域的主要挑战是开发适当的基因变异操作并找到合适的预言者。

                                   八。          R ELATED W ORK

在机器学习领域已经有多篇关于对分类器的规避行为的攻击，主要集中在简单模型的垃圾邮件检测上[6,10,18]，Chinavle等人认为对手问题本质上是*概念漂移*，是考虑到数据分布随时间变化的机器学习领域[6]，然而，概念漂移解决方案假设数据分布的变化不是由于分类器本身造成的，而是由对手故意适应的。

Biggio等人以前已经研究过针对恶意软件分类器的逃避攻击。从分类模型[4]和Srndic et al。[28]。然而，这些研究假设攻击者只能插入新的特征，他们在特征空间中进行了逃避实验而没有产生实际的逃避PDF恶意软件。实际上，我们工作的实验表明，攻击者也可以在保留恶意的同时删除特征，而且我们的实验证实，由此产生的躲避变种通过在测试环境中的动态执行来保存恶意。

Srndic等人 展示了如何通过利用特征提取中的实现缺陷来避免PDFrate [29]。我们的方法不依赖目标分类器中的任何特定实现缺陷。相反，它利用分类器模型特征空间中的弱点，并采用随机方法以不同的方式处理样本。

Maiorca等人 提出了针对PDF恶意软件分类器的反向模仿攻击[20]。在反向模仿中，通过将恶意有效载荷插入到结构中，良性样本被操纵为恶意样本。这种攻击是基于结构特征的一类分类器通用的。但是，手工攻击仅适用于具有简单有效载荷的恶意软件。相反，我们基于GP的方法是自动的，并没有这个限制。

演化算法最近也被用来欺骗基于深度学习的计算机视觉模型[22]。相反，这项工作使用遗传编程，这是进化算法的一个重要分支，用于生成计算机程序等高度结构化的数据。

                                      IX。        ç 面上污染物

我们的实验显示了如何构建机器学习分类器的传统方法可以对决定的对手失败。我们认为，安全应用中使用的分类器的设计者必须考虑对手如何适应这些分类器，这对于研究团队开发更好的方法来预测分类器在部署中的实际有效性是重要的。

一个VAILABILITY

我们的自动逃避工具的源代码以及我们所有实验的数据可以在http://www.EvadeML.org上找到。

一个CKNOWLEDGMENTS

这项工作部分得到了空军科学研究办公室和国家科学基金会的资助。作者非常感谢Giovanni Vigna提供了有益的帮助，Westley Weimer就这项工作进行了富有成果的讨论，并帮助进行了遗传编程实验，王宏宁提供了深刻的意见和帮助。我们感谢Nedim Srndic和Pavel Laskov制作PDFrate的开源版本并提供Hidost，这对我们的实验是必不可少的。

R EFERENCES

[1]       Adobe，Inc. PDF参考和PDF扩展到PDF规范。http://www.adobe.com/devnet/pdf/ pdf reference.html 。

[2]       Marco Barreno，Blaine Nelson，Russell Sears，Anthony D Joseph和J Doug Tygar。机器学习是否安全？在*第一ACM研讨会信息，计算机和通信安全（ASIACCS）*，2006年。

[3]       巴蒂斯塔·比吉奥，乔治·弗梅拉和法比奥·罗里。对抗分类任务的多分类器系统。在*多分类系统*。施普林格，2009年。

[4]       巴蒂斯塔·比吉奥，伊基诺科罗纳，达维德·马奥卡，布莱恩·尼尔森，内迪姆·斯隆迪，帕维尔·拉斯科夫，乔治·贾科托和法比奥·罗利。在测试时间对逃避机器学习的攻击。在*6 日欧洲机器学习和数据挖掘大会（ECML / PKDD）*。2013。

[5]       Stephan Chenette。恶意文档存档的签名测试和研究 - Contagio恶意软件转储。http://contagiodump.blogspot.de/2010/08/ malicious-documents-archive-for.html 。

[6]       迪帕克Chinavle，Pranam科拉里，添Oates的，和蒂姆·芬。垃圾邮件的对抗分类合奏。在*18 届 ACM会议上的信息和知识管理（CIKM）* 2009年，。

[7]       Marco Cova，Christopher Kruegel和Giovanni Vigna。驱动下载攻击和恶意JavaScript代码的检测和分析。在*第十九届国际万维网会议（WWW）*，2010。

[8]       CVE详细信息。Adobe Acrobat Reader - CVE安全漏洞，版本和详细报告。http：//www.cvedetails.com/product/497 。

[9]       乔治·ê·达尔，杰克W¯¯斯托克斯，丽登，董羽。使用随机投影和神经网络的大规模恶意软件分类。在*第 38 届IEEE声学，语音和信号处理（ICASSP）国际会议上*，2013。

[10]    Nilesh Dalvi，Pedro Domingos，Sumit Sanghai和Deepak Verma。敌对分类。2004 *年第 10 届 ACM SIGKDD国际知识发现与数据挖掘会议（KDD）*。

[11]    斯蒂芬妮·福雷斯特。遗传算法：自然选择原理在计算中的应用。*科学*，261（5123），1993。

[12]    克劳迪奥Guarnieri，Alessandro Tanasi，Jurriaan Bremer和Mark Schloesser。布谷鸟沙箱：恶意软件

分析系统。http://www.cuckoosandbox.org/ 。

[13]    马克哈曼，威廉乙兰登，和韦斯特利韦默。遗传编程的逆向工程。在*20 日于逆向工程的IEEE工作会议（WCRE）*，2013。

[14]    Thomas Hungenberg和Matthias Eckert。INetSim：Internet服务仿真套件。http://www.inetsim.org/ 。

[15]    约翰R柯扎。*遗传编程：在自然选择的计算机编程*，第1卷。麻省理工出版社，1992年。

[16]    Pavel Laskov和Nedim Srndi c。“静态检测恶意的支持JavaScript的PDF文档”。在*27 日*

*ACM年度计算机安全应用会议（ACSAC）*，2011。

[17]    克莱尔勒Goues，ThanhVu阮，斯蒂芬妮福雷斯特，和韦斯特利韦默。GenProg：一种通用方法

用于自动软件修复。*IEEE Transactions on Software Engineering*，2012。

[18]    Daniel Lowd和Christopher Meek。对抗学习。2005 *年第 11 届 ACM SIGKDD国际知识发现与数据挖掘会议（KDD）*。

[19]    Davide Maiorca，Giorgio Giacinto和Igino Corona。恶意PDF文件检测模式识别系统。在*8 个机器学习和数据挖掘技术在模式识别国际会议*。2012。

[20]    大卫·马奥卡，Igino Corona和Giorgio Giacinto。看袋子还不够发现炸弹：恶意PDF文件检测结构方法的一种回避。在*8 日 ACM研讨会信息，计算机和通信安全（ASIACCS）*，2013。

[21]    Patrick Maupin。PDFRW：读取和写入PDF的纯Python库。https://github.com/pmaupin/ pdfrw 。

[22]    映阮，杰森Yosinski，和Jeff Clune。深度神经网络容易被愚弄：对不可识别图像的高置信度预测。在*第 28 届 IEEE计算机视觉与模式识别（CVPR）会议上*，2015。

[23]    Conor Ryan。*使用遗传编程软件的自动重新设计*，第2卷。Springer Science＆Business Media，2012。

[24]    KARTHIK Selvaraj和尼诺佛瑞德Gutierrez的。PDF恶意软件的崛起。https://www.symantec.com/content/en/ us / enterprise / media / security response / whitepapers / pdf malware.pdf的崛起.pdf ，2010年3月。

[25]    查尔斯Smutz和安耶洛斯Stavrou。使用元数据和结构特征进行恶意PDF检测。技术报告，2012。

[26]    查尔斯Smutz和安耶洛斯Stavrou。使用元数据和结构特征进行恶意PDF检测。在*28 届 ACM年度计算机安全应用大会（ACSAC）*，2012。

[27]    内迪姆Srndic和Pavel Laskov。Mimicus：对抗分类器逃避图书馆。https://github.com/srndic/ mimicus 。

[28]    内迪姆Srndic和Pavel Laskov。基于分层文档结构的恶意Pdf文件检测。在2013 *年的第 20 届网络和分布式系统安全研讨会（NDSS）上*。

[29]    内迪姆Srndic和Pavel Laskov。基于学习的分类器的实际规避：一个案例研究。在*第 35 届 IEEE安全与隐私专题研讨会（奥克兰）上*，2014。

[30]    赛门铁克公司。赛门铁克互联网安全威胁报告，2015。

[31]    VirusTotal。免费的在线病毒，恶意软件和URL扫描器。https://www.virustotal.com/ 。

一个PPENDIX

*A. PDFrate中的突变特征*

表6列出了在我们的实验中突变的68个特征。然而，重要的是要注意，仅仅因为这里没有出现一个特征，并不意味着这个特征对于规避是有效的。列出的特征是那些足以在我们的实验中达到100％逃避率的特征。类似地，单向突变是如何在回避攻击实验中观察到的。这并不一定意味着这些功能也不能在相反的方向上发生变异，而不会在实践中破坏恶意软件样本。

            表六。               68°F 功能设定中号ODIFIED Ë VADING PDF RATE

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 可变性 | 功能名称 | 可变性 |
| 箱子其他类型箱子其他只计算acroform计数动作  计数盒子字母计数盒其他计数endobj  计数终端计数字体  计数图像医学  计数图像小计数图像总数  计数图像xsmall计数javascript计数js  计数obj  计数objstm计数页面  计数页面obs  计数流  创建不匹配创建tsE:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image032.gif  创造tz  创建版本比率创建者点创建者lc  创造者len  创建者不匹配创建者数量创建者oth创建者uc delta ts  三角洲tz  图像不匹配 | ＃  “  “＃  “＃  ＃  “＃  “＃  “＃  “＃  “  “  “  “  ＃  ＃  “＃  ＃  “＃  ＃  “＃  “  “  ＃  “  “＃  “＃  “＃  “  “＃  “＃  “＃  “  ＃ | 图片totalpx len obj avg  len obj max len obj min  len流平均  len流最大len流最小  pos acroform avg  pos acroform max  pos acroform分钟pos盒子avg  邮箱最大邮箱最小邮寄平均价格  最大值最小值  pos图像平均  E:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image033.gifpos图像最大pos图像最小pos页面avg  pos页面最大值页面最小生产者点生产者lcE:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image032.gif  生产者len  生产者不匹配生产者数量生产者oth  生产者uc  比例imagepx 大小比例大小objE:\bdfirst\2018\2018Q1\Automatically Evading Classifiers A Case Study on PDF Malware Classifiers.files\image032.gif  比率大小页面  比例大小的流  尺寸 | “  “＃  “＃  “＃  “＃  “  “＃  “＃  “＃  “＃  “＃  “＃  “＃  “  “  “  “  “  “  “＃  “＃  “＃  “＃  “＃  “＃  “  “＃  “＃  “＃  “＃  “＃  “＃  “＃ |

                                                   “                                                            ”

*B. Hidost中的突变特征*

表VII中列出了24个插入特征和19个删除特征，用于查找针对Hidost的2,859个回避变体。与PDFrate一样，没有列出的功能不一定是强大的功能。

“计数”是突变该特征的回避变体的数量。请注意，某些功能在PDF对象结构中是分层依赖的，因此一个插入或删除可能会影响许多功能。例如，插入一个完整的*元数据*对象（就像2,507个变体中那样）也引入了几个子对象：*元数据/长度*， *元数据/子类型*和*元数据/类型*。

                表VII。               F EATURES A LTERED E VADING H IDOST

|  |  |
| --- | --- |
| 计数 | 插入的功能 |
| 2507 | 元数据 |
| 2507 | 元/长度 |
| 2507 | 元/亚型 |
| 2507 | 元/类型 |
| 2,454 | PageLabels |
| 2363 | ViewerPreferences /方向 |
| 1991 | 页/资源/ ProcSet |
| 1968 | 页/资源 |
| 1702 | 页/旋转 |
| 1382 | 页/媒体框 |
| 825 | 主题 |
| 718 | OpenAction /媒体框 |
| 385 | OpenAction /内容/过滤器 |
| 385 | OpenAction /内容/长度 |
| 369 | OpenAction /目录 |
| 319 | OpenAction /资源 |
| 319 | OpenAction /资源/ ProcSet |
| 158 | OpenAction /旋转 |
| 158 | OpenAction /裁剪框 |
| 51 | OpenAction /类型 |
| 51 | OpenAction |
| 41 | PageLabels /订购数量 |
| 41 | PageLabels /订购数量/ S |
| 40 | 页面布局 |
| 计数 | 已删除的功能 |
| 1345 | 名称/ JavaScript的/姓名/ S |
| 865 | 页面布局 |
| 615 | 大纲/类型 |
| 615 | 纲要 |
| 615 | 概括/计数 |
| 502 | AcroForm /场 |
| 500 | AcroForm |
| 330 | OpenAction / JS /长度 |
| 54 | 页/旋转 |
| 14 | 页/资源/ ProcSet |
| 12 | AcroForm / DR /编码/ PDFDocEncoding的 |
| 12 | AcroForm / DR /编码/ PDFDocEncoding的/差异 |
| 12 | AcroForm / DR /编码/ PDFDocEncoding的/类型 |
| 11 | 页/资源 |
| 9 | AcroForm / DA |
| 8 | 页/媒体框 |
| 4 | OpenAction / S |
| 3 | 名称/ EmbeddedFiles |
| 2 | 名称 |

[[1]](file:///E:\\bdfirst\\2018\\2018Q1\\Automatically%20Evading%20Classifiers%20A%20Case%20Study%20on%20PDF%20Malware%20Classifiers.htm" \l "_ftnref1" \o ") Mimicus作者无法在Contagio中找到MD5散列35b621f1065b7c6ebebacb9a785b6d69的恶意文件。

[[2]](file:///E:\\bdfirst\\2018\\2018Q1\\Automatically%20Evading%20Classifiers%20A%20Case%20Study%20on%20PDF%20Malware%20Classifiers.htm" \l "_ftnref2" \o ")具体而言，Hidost作者声称：“我们可以设想该最积极的逃避策略是成功的仅0.025％的针对与使用二进制嵌入RBF核断开的，现成的非线性SVM分类测试恶意例子。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，只与分类器使用的真实特征松散相关。真实特征的空间隐藏在一个复杂的非线性变换之后，这个变换在数学上很难逆转。“[28]