《基于统计学习的人工智能系统的毒饵攻击技术》

项目研究成果

基于统计学习的神经网络模型的毒饵攻击策略

郭帅[1]，李关玲[1]，胡琪[1]，陈城义[1]，王志[1]
(1. 南开大学 网络空间安全学院 ,天津 300350）

摘要：本文提出一种基于统计学习的针对神经网络的毒饵攻击技术。以二分类模型为攻击目标，结合了对抗攻击和投毒攻击策略的优点，通过本地对抗模型提取恶意样本关键特征，减少了攻击者对目标模型的访问次数；在对抗模型的基础上引入不一致性度量参数 P-vale，以统计的方法确定边缘良性样本。利用边界良性样本参与训练后对模型边界参数的影响能力，插入恶意样本的关键特征制作毒饵。少量的投毒即可生成一批针对性后门，模型在干净数据集上的表现几乎不受影响。

关键字：投毒攻击；神经网络后门；对抗性攻击；恶意代码识别

1. 介绍

1.1 背景分析

神经网络因为其处理大数据和模式识别的能力而被大范围应用。神经网络依赖数据，其复杂度主要来自于数据，通过训练数据，网络获得了分类和预测的能力。数据是神经网络学习、判断、行动的唯一标准，而数据的价值在于其完整性、隐私性和可用性。在攻击神经网络时，攻击者通过多次交互操纵样本数据的完整性绕过模型决策边界从而欺骗模型，或是通过在训练数据中加入有毒数据改变目标模型，进而通过分类检测。

随着神经网络作为产品被部署在各个领域进行决策，以上攻击方法带来的威胁也愈加突出。作为产品，无论是更新升级自己或者是提供服务，神经网络都需要和用户进行交互。而在交互和模型的更新训练过程，存在数据被污染和模型被泄露的风险。由于神经网络不透明[错误!未找到引用源。]，一旦其训练数据被污染，管理者很难第一时间发现问题，从而造成严重后果。

人工智能深度融入社会，其安全性要求提高。提前发现威胁可以帮助机器学习模型变得更安全，增强 AI 决策的可信度，从而加速各类 AI 项目的落地。

1.2 相关工作

神经网络发展至今，流行的神经网络主要有前馈神经网络、卷积神经网络和循环神经网络。卷积神经网络具有权值共享的特性，可以很好地处理包含空间连续性的输入[12]，通常应用于图像处理。循环神经网络具有记忆性，适合处理具有时间上的连续性的输入数据，常应用于自然语言处理。

1.2.1 本文选择的目标神经网络

考虑到神经网络攻击中有关文件分类的研究较少，Android 系统应用分类检测具备现实需求，其充足的 apk 样本量也可以满足模型训练的需要，本文的目标模型设置为恶意代码分类器。恶意代码的特征通常是直接从目标程序中提取出来，它的样本特征不具有以上两种神经网络所处理特征的属性[5][6]。而前馈神经网络可以满足代码分类要求的准确率，所以本文采用简单的前馈神经网络用于安卓恶意代码分类检测。

1.2.2 现有的神经网络攻击方法

在机器学习领域常见的攻击分为二种：对抗攻击与投毒攻击。

对抗攻击不改变目标模型，而是不断修改恶意样本使其绕过模型的分类边界，从而骗过神经网络，要求攻击者可以和目标模型进行互动。这类样本一般处在模型训练所得边界与真实边界之间，即对抗区域。攻击者通过生成对抗样本进行对抗攻击，可分为白盒和黑盒攻击。白盒攻击者拥有目标模型，可以互动，熟知参数和算法；黑盒环境下目标模型是一个输入机，攻击者可以观察输入对应的输出，获取模型信息，根据获取的信息构建一个与目标模型接近的子模型，用子模型制作对抗样本来对目标模型进行攻击。因为对抗样本具有可迁移性，所以能够使本地子模型分类错误的样本也可以适用于目标模型。相较于白盒环境下的攻击，黑盒环境的攻击难度更大，对攻击者的技术要求很高。最近几年的黑盒攻击方法有：One-pixel[7]， UPSET[8]，ANGRI[8]，Houdini[9]，等。

对抗攻击的防御策略，根据防御发生的时间，可以分为被动型和主动型防御。前者在构建完机器学习系统之后借助辅助网络检查对抗样本，后者在训练阶段引入对抗样本，使得生成的模型更健壮。这些方法被证明防御效果并不明显。

投毒攻击改变目标模型，使模型产生漏洞，要求攻击者能够操控网络的训练集。可进一步分为无差别投毒攻击、针对性投毒攻击和神经网络后门。无差别攻击的目的是破坏模型的可用性，通过更改数据标签或者伪造数据，模型学习到错误的信息，进而产生错误判断。针对性投毒会将原始样本中的一些特征提取出来，插入到目标标签样本中。机器学习模型学习了这些样本后，这些特征会被认为是属于目标类，有该特征的样本就会被识别为目标类。攻击会使模型边界发生偏移，但模型本身的改动不会很大。需要注意的是，这种方法不需要改变样本的标签。神经网络后门具有高度隐蔽性，攻击者将一类样本（A 类）进行针对性的修改或者插入特定的标志（m）后，修改它的标签，将其伪装成另一类样本（B 类）。毒化的数据被网络接收后，会导致神经网络产生后门。后门就是这些特征，数据在不加这些特征时模型的表现良好，加入这些特征后模型分类就会产生错误。

1.2.3 现有的攻击方法缺陷

以往的对抗性攻击的研究集中在白盒环境。将攻击方法应用于黑盒环境下时，由于原始特征集的缺失，攻击效果会有所下降。有些学者提出通过偷取模型的方法来提升攻击效果，但偷取模型需要攻击者在目标模型上进行大量的查询来获得反馈。

现有的投毒方法也存在问题。无差别投毒会破坏模型的可用性，由于模型对于数据分类的准确率会大幅下降，在对模型进行测试的时候很容易被检测出来。针对性投毒攻击隐蔽性较强，但是通常只能针对特定的样本，不能通用。而神经网络后门方法需要攻击者具有控制样本标签的能力，对攻击者能力要求高。

神经网络后门的研究主要集中在图像分类领域。在图像分类时，由于没有可信的自动化筛选机制，我们可以选择控制标签的神经网络后门方法[1][2][3]对其进行攻击。但是在文件分类领域，存在像 Virus Total 这样的网站可以对于样本进行重新标记。即使攻击者将恶意样本标记为良性样本，人们借助第三方平台也可以将被标记为良性的恶意样本筛选出来。投毒样本容易在训练前被剔除。

1.2.4 本文贡献

解决以上攻击方法的缺陷，提出一种适用于文件分类模型的全新攻击策略。该策略在黑盒环境下也可适用，不需要控制标签，便可以对特征进行针对性的毒化。基于本地对抗模型发现关键特征，通过统计分析找到低 P-value[4]的边缘样本，只需与目标模型的少量交互便可达到最优的攻击效果。

2. 系统实现

2.1 系统方案

本系统在预先对子模型进行特征筛选获取关键特征的基础上，利用统计学习计算出的低P-value样本使得目标模型的决策方式更容易被发现。通过对抗模型的建立，在面临一个灰盒或是黑盒模型时，可通过本地子模型预先获取足够多的信息，减少对查询次数的要求，同时提高攻击效果。因为对抗样本具有可迁移性，所以在本地子模型分类边界建立时权重最大的样本也可以影响目标模型。

考虑到模型为尽可能分类不同样本，尤其是正确预测边界样本（离目标类簇远而离决策边界近的样本）时，会对模型权重进行较大的调整。引入不一致性分数度量参数 P-value，通过计算子模型良性样本集合的 P-value 统计值，获取 P-value 最低的良性样本，也就是远离目标类簇的良性样本。在本地对抗模型上我们通过特征筛选算法提取恶意样本关键特征，插入低 P-value 的良性样本中，制作用于攻击的毒饵。当我们将毒饵投入目标模型后，模型为了更好的分类这个样本，便会对样本权重进行改变，离目标类簇越远模型的改变越大，同时模型为不影响在其他样本上的准确率只会对我们提取的恶意样本的特定特征进行更改。这些特征便是目标模型上生成的后门。通过计算躲避率检测模型是否生成了神经网络后门。

2.2 系统架构

本攻击技术基于 Python 语言开发，主要包括计算边缘样本、提取关键特征、实现毒饵攻击三个部分。

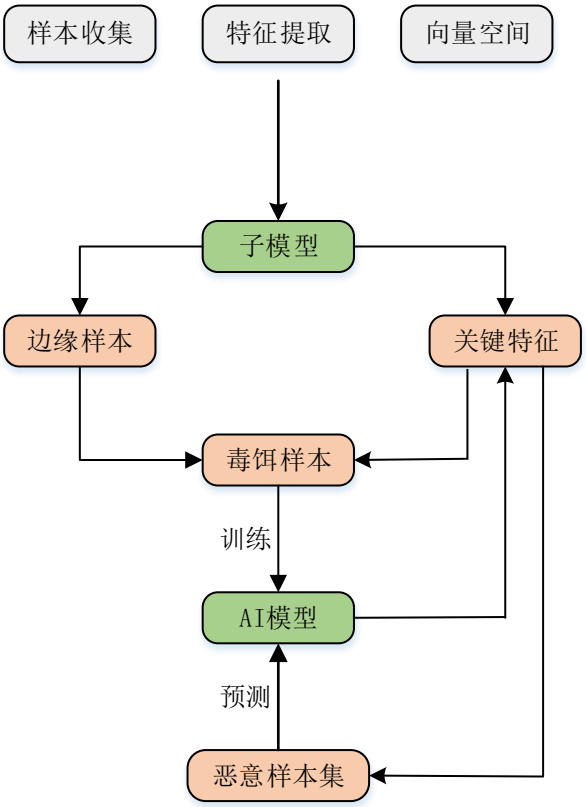

图 2.1 系统架构图

样本向量化的过程，如图 2.2 所示。根据 Drebin[11]中的提取方法，从安卓应用的清单文件中提取特征，相应的维度设为 1，其余维度设为 0。

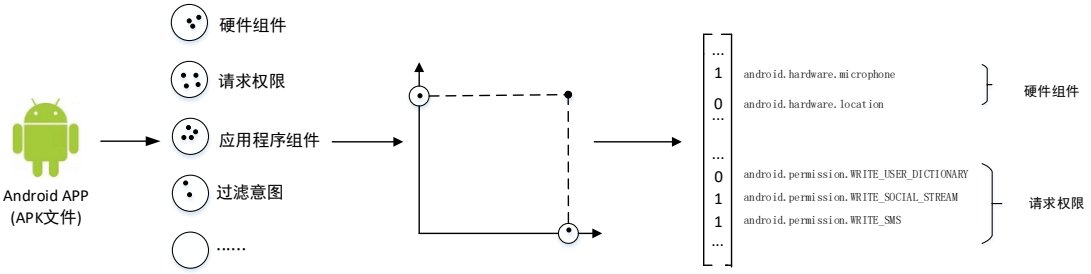

图 2.2 样本向量化过程

子模型的训练采取三层前馈神经网络。基于子模型和统计分析方法发现边缘样本和提取关键特征（见 2.3 节）后，通过脚本调用 apktool.jar、sigapk.jar 等开源工具，修改良性 apk 文件的 Manifest.xml 清单文件制作毒饵。

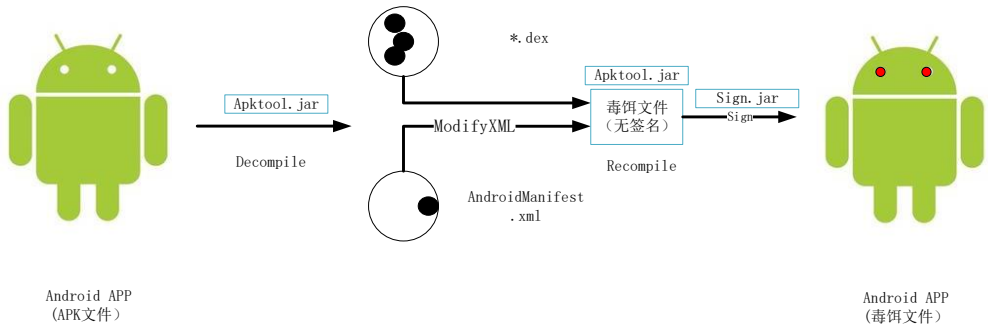

图 2.3 制作毒饵文件过程

2.3 系统核心技术

2.3.1 基于统计学习计算边缘样本

基于统计学习的人工智能系统毒饵攻击技术中，为了实现算法的改进，增加投毒攻击的成功率，我们在机器学习的基础上用统计学习计算不一致性度量参数 P-value，筛选出最容易使模型权重发生变化的边缘样本。这种攻击技术改变了传统投毒攻击使用随机样本作为毒饵的攻击模式，引入对概念漂移更敏感的 Conformal Prediction[10]统计学习算法，计算统计量 P-value 过程如图 2.4。

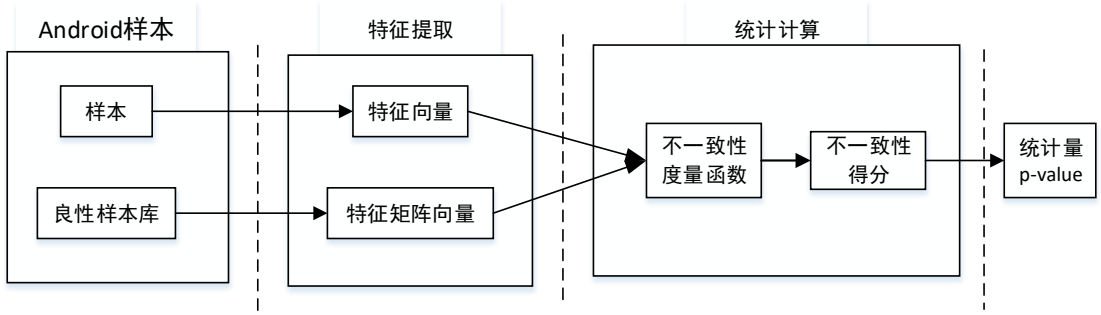

图 2.4 统计量 P-value 计算过程

P-value反映了一个样本与已知样本集合在统计规律上的一致性。它可以对任何基于阈值的分类或聚类检测算法的预测结果进行评估，并基于用户所能接受的最大误差程度来预测样本属于某一样本集合的可信度。

P-value定义和计算过程如下：

设X为待测量未知向量；良性样本特征矩阵为D($x_1, x_2, x_3 \ldots x_n$)，良性样本特征矩阵可重复但无序，多为已被分类或聚类算法后处理的具有某种相似性的良性样本特征向量的集合。定义A为不一致性度量函数，如神经网络这种能够表示不一致性的函数：

$$A(X, D) = s \tag{9}$$

把需要检测的特征向量V放入良性特征矩阵D中，构成新的矩阵D'；逐步从D'中取出特征向

量X_i($1 \le i \le n+1$)，通过不一致性度量函数计算向量X_i与取出X_i后的矩阵的不一致性得分s_i：

$$A(X_i, D' - X_i) = s_i \tag{10}$$

计算出所有的n+1个向量的不一致性得分，样本向量X与特征矩阵D的不一致性得分为s_{n+1}；

当A不一致性度量函数为距离函数时，不一致性得分表示样本向量X与良性样本矩阵D的距离，统计所有不一致性得分大于等于s_{n+1}的特征向量的个数M:

$$if\ s_i \ge s_{n+1} \quad \to M \quad for\ i \in \{0,1,...,n+1\} \tag{11}$$

当A不一致性度量函数为相似性函数时，不一致性得分表示样本向量X与良性样本矩阵D的相似程度，统计所有不一致性得分小于s_{n+1}的特征向量的个数N:

$$if\ s_i < s_{n+1} \quad \to N \quad for\ i \in \{0,1,...,n+1\} \tag{12}$$

特征向量X相对于良性样本矩阵D的统计量:

$$p - value = M / (N+1) \tag{13}$$

2.3.2 优化的特征筛选算法提取关键特征

一个大型的神经网络，其权值和偏重的数目往往数以万计。神经网络的数据集包含几百万个特征，其中冗余特征对模型的性能几乎没有帮助，甚至有些被列为噪声的特征还会降低模型的性能。随机选取的特征很可能在目标模型中根本不存在。因此，为了提高投毒攻击的准确率，我们使用了梯度排序算法和特征筛选算法来筛选出高信息熵的特征。

2.3.2.1 通用恶意代码躲避算法

在进行对抗攻击和投毒攻击时，我们通常会使用通用恶意代码躲避算法来找到最容易使恶意样本被认为是良性样本的特征。设恶意样本集为X($1,2,...,n$)，分类器为f，定义$f(x_i)$为计算样本往良性类方向的梯度：

$$f(x_i) = \nabla f_t(x_i) \ \ for\ i \in \{1,2,...,n\} \tag{6}$$

定义特征集为V：

$$V = V + \nabla f_t(x_i) \tag{7}$$

最后对V中特征按其值升序排序。

具体过程如算法2.4。

算法2.4 通用恶意代码躲避算法

输入：恶意样本集X,分类器f

输出：根据平均梯度排序后的特征

初始化：V=0

1.repeat

2. 对$f(x_i)$计算往良性类方向的梯度$\nabla f_t(x_i)$

3. $V = V + \nabla f_t(x_i)$

4.until 遍历样本集 X

5.对V根据其值升序排序

显然该算法需要计算分类器往良性类方向的梯度，在本地子模型中是适用的。然而在黑盒模式下，攻击者处于对模型完全未知的状态，既无法拿到目标模型，也得不到任何反馈。因此在黑盒模式下，攻击者在子模型中计算出的最低梯度特征在目标模型中攻击的效果并不好。在黑盒模式下，我们设计的改进方案是先计算出模型的低 P-value 良性样本，然后将通过梯度排序算法计算出的恶意样本集的最低梯度特征插入边缘良性样本中，并用其对模型进行训练，以修改模型的权值和参数。

2.3.2.2 特征筛选算法

通常目标模型出于安全防御的考虑并不会返回概率，而是只返回真实标签。特征筛选算法就是基于攻击者无法得到目标模型，也得不到模型所反馈的概率，但能获取少量标签反馈的基础上设计的。

计算出恶意样本的 P-value,设低 P-value 恶意样本集为 X($x_1, x_2, ..., x_m$)，使用算法 2.1 得到梯度排序的前 n 个特征设为 Y($y_1, y_2, ..., y_n$)，设 f 为目标模型。 定义特征集合 V，对 Y 中每个特征 y_i（$1 \le i \le n$）将其插入样本 x_j（$1 \le j \le m$）中构成样本 x_j^*,用模型 f 对样本进行 x_j^* 检测：

$$V[i]{+}{+} \quad if\ f(x_j^*) == \text{良性}\ \ for\ j \in \{1, 2, ..., m\} \tag{8}$$

最后对 V 按其值降序排序。

具体过程如算法 2.5。

算法 2.5 特征筛选算法

输入：低 P-value 的恶意样本集 X，根据梯度排序的前|Y|个特征 Y，f为目标模型

输出：重新排序后的特征

初始化：V=0

```
repeat
    repeat
        将特征 y_i 插入 x_j 个样本中
        if (f(x_j^*)==良性):
            V[i]++
    until 遍历样本集合 X
until 遍历特征集 Y
对V根据其值降序排序
```

通过通用恶意代码躲避算法，我们已经计算出恶意样本集中最容易被模型分类为良性样本的特征排序，我们需要注意这些特征是在本地子模型上得出的。而使用低 P-value 恶意样本的原因是目标类簇的边缘样本被错误分类的可能性更大。通过特征筛选算法，我们提取出边缘恶意样本被目标模型错误分类为良性的关键特征。如果低 P-value 恶意样本集的大小为 m，梯度特征排序后我们选取前 n 个特征，则总共需要查询目标模型 m·n 次，时间复杂度为O(mn)。

3.创新点说明

3.1 针对文件

传统的攻击主要是针对图片，图像分类中没有可信的自动化筛选机制，可实行控制标签的神经网络后门攻击，但是文件分类中存在可信平台，修改的标签会被重新检测出。因此，我们不去控制标签，只是在良性样本的 Manifest 文件中插入恶意样本特征，该样本与正常样本无任何不同，能够正常运行，也可以通过可信平台和目标模型的检测。而该类加注了关键特征的良性样本，作为毒饵，在投入目标模型后会产生后门。降低攻击难度的同时，为后门的生成提供了可能性。

3.2 改进现有攻击方法

考虑到现有攻击方法存在的各种问题，本项目提出了新的攻击方法。

3.2.1 对于对抗性攻击的改进

我们利用离边界近的样本来对特征进行筛选。攻击者选择少量在子模型中离边界近的样本，当某个特征能导致多个样本改变标签，则说明该特征在目标模型中依然有很低的梯度。

3.2.2 对于数据投毒攻击的改进

改变样本标签的方法对于攻击者的技术有一定要求，并且存在 Virus Total 这样的平台可以对于恶意代码就进行重新标记。为了降低攻击者的攻击难度，我们提出了一种不控制样本标签的投毒策略，该策略同样可以达到与控制标签相同效果。

我们制作有毒样本的方法为：将特征插入离恶意样本近的良性样本中。这种策略的依据是模型总是会尽力把不同类别的样本区分开（恶意样本和良性样本），当我们将特征（来自恶意样本）插入离恶意样本近的良性样本时，模型为了更好地分类这个样本，会对模型的权重进行改变。此时，我们不需要控制样本标签，因为我们是使用良性样本制作的有毒样本，它本就属于目标类（良性样本类）。

3.3 引入统计参数 P-value

为了支持以上两种改进算法，即得到离边界近的样本，我们引入了不一致性度量参数 P-value。P-value 表示为样本的不一致性分值在一个数据集合中的百分位，值域在 1 到 0 之间，以统计的方法表现样本与数据集的相似性。预先通过子模型进行了特征筛选，又利用低 p-value 样本使得目标模型的决策方式更容易被发现。所以我们可以通过极少量的查询次数达到最优的攻击效果。

3.4 实现有毒样本的自动制作

为了对神经网络模型进行攻击，我们需要制作有毒样本，我们的方法是将良性样本的关键特征插入低 P-value 的良性样本，来让模型倾斜等等。为此，我们通过编程，实现了样本文件的自动修改来生成有毒样本。

4. 测试与分析

本次实验采用 Drebin 项目的公开数据集 。选取了 2014 年、2015 年共 20 万条数据训练目标模型。

图4.1 Drebin部分数据集截图

经过测试，我们提出的改进的攻击策略对于目标模型的攻击能够达到很好的效果。在白盒、黑盒、灰盒情况下，我们的攻击策略都能以大概率使恶意样本成功被目标模型识别为良性样本。我们的策略具有隐蔽性，高效性，低要求性，是一种可信赖的攻击方法。

4.1 测试方案

恶意样本及良性样本标签借助 Virus Total 标记。如果出现两个以上引擎判定这个 APK 是恶意的，我们就标记为恶意。如果没有引擎判定其为恶意，我们则标记为良性。

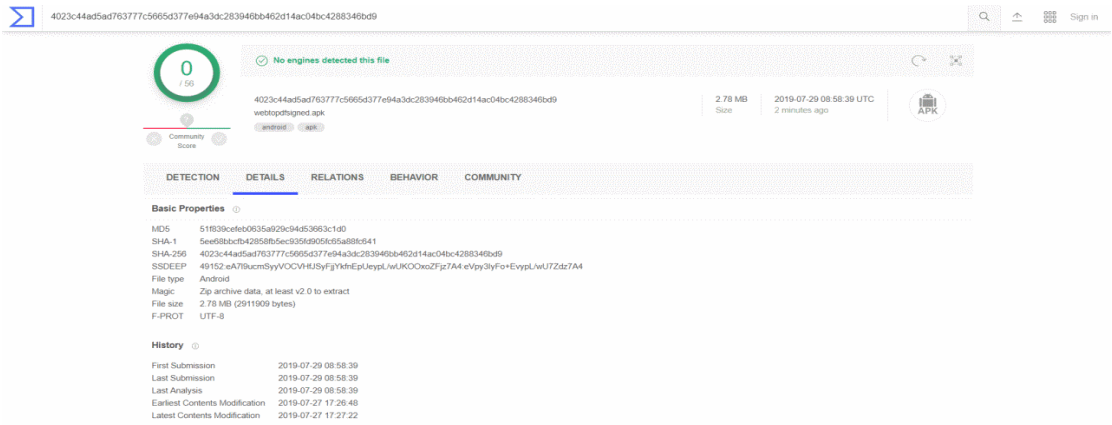

图4.2 virus total报告图

在神经网络中，投毒攻击分两种情况，重训练与模型更新。令D_T为训练集，D_A为投毒样本集，M_1为目标模型 1，M_2为目标模型 2。

重训练：$\{D_T = D_T \cup D_A\} \xrightarrow{train} M_1$

模型更新：$\frac{1}{2}D_T \xrightarrow{pre-train} M_{pre}$

$$\{M_{pre}, (D_T - \frac{1}{2}D_T) + D_A\} \xrightarrow{update} M_2$$

目标模型 1：将训练集分为 20 万良性样本，20 万恶意样本；测试集为 3 万良性样本，3 万恶意样本。训练集包含特征 267 万，选择 Adam 学习器，0.001 的学习率，2000 样本为一批，训练 5 轮选择效果最好的模型。基于训练集建立模型 M1 并在测试集上经测试得到准确

率：

目标模型 2：将训练集分为 10 万良性样本，10 万恶意样本；更新集为 10 万良性样本，10 万恶意样本；测试集为 3 万良性样本，3 万恶意样本。训练集包含特征 155 万，2000 样本为一批，训练 5 轮选择效果最好的模型。基于训练集建立模型 M2 且在测试集上经测试得到准确率：

为了模拟真实的攻击情况，我们假设攻击者有 1%的恶意样本，也就是 2000 个恶意样本。而良性样本是可以从网络上爬下来的。根据我们的假设，攻击者投毒时需要低 P-value 的良性样本。所以我们做了几组相关实验来验证子模型中的低 P-value 样本是否在目标模型中仍然是低 P-value 的。

实验一：2000 恶意样本和 2000 良性样本建立模型；

实验二：2000 恶意样本和 10000 良性样本建立模型；

实验三：2000 恶意样本和 50000 良性样本建立模型。

我们令 X 轴代表样本在子模型中的 P-value 值，Y 轴代表样本在目标模型中的 P-value 值，画出他们在模型中的 P-value 分布，左为整体分布，右为子模型中 P-value 最小的 100 个样本的分布。如图 3.3。

预期的实验结果是，子模型中 P-value 低的样本在目标模型中的 P-value 也很低，且良性样本越多，所能拿到的样本的 P-value 越低。攻击者完全可以通过子模型来得到所需的低 P-value 样本。实验证明确实如此。

接下来的实验，我们取平均情况，假设攻击者有 2000 恶意样本和 10000 良性样本用来建立子模型，不失一般性，我们用三组不同的数据建立了三个子模型。通过子模型，我们得到了所有用来投毒的低 P-value 样本。

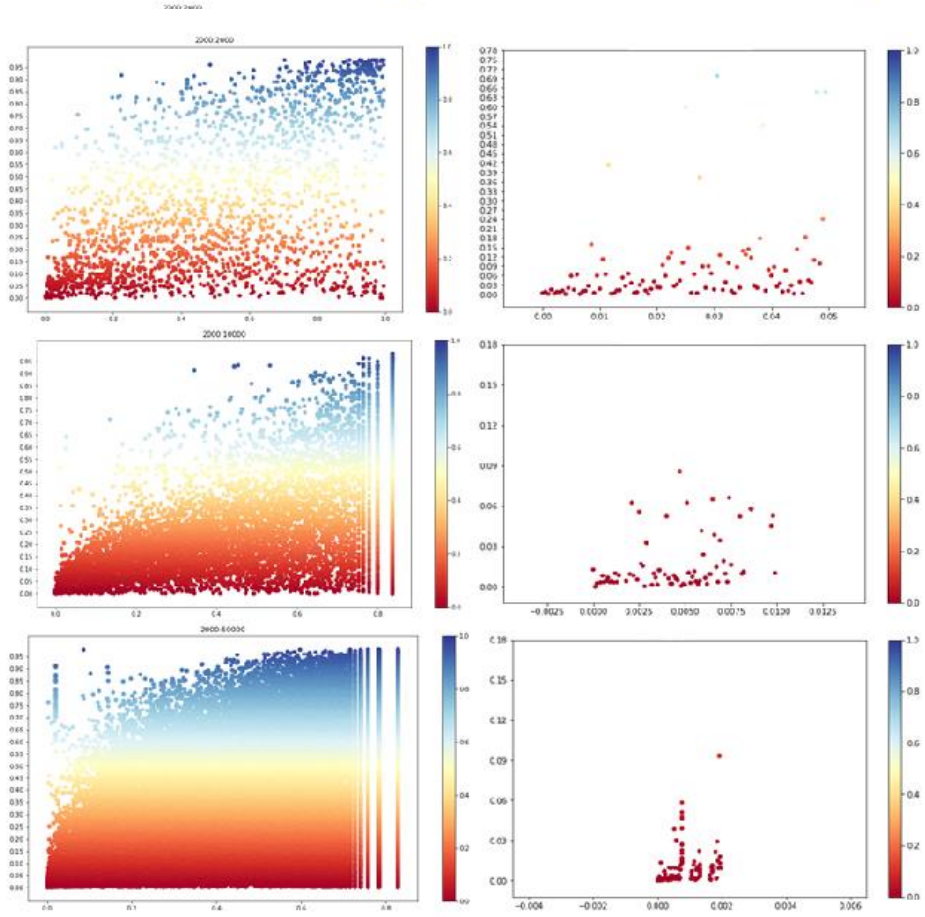

图 4.3　P-value 分布

我们在测试时对算法和功能进行了测试：

算法测试方面，我们进行了两组对比试验。第一组是用通用恶意代码躲避算法筛选的特征与用改进的算法筛选的特征进行投毒来作对比。第二组是用随机样本与低 p-value 样本进行投毒作对比。

功能测试方面，为了验证在真实场景下投毒攻击的有效性，我们的实验主要基于黑盒环境，并且基于更新与重训练的两种目标模型。利用三个不同子模型，对目标模型进行投毒测试，计算躲避率。为了证明攻击的危害性，我们仅投毒大约 0.01%的样本，也就是 40 个。

4.2 测试环境及设备

本文的实验系统为 Ubuntu16.04 Long Term Support，实验中主要的语言为 Python。部署该模型系统需要 Python 环境和 Jre1.8。本文所有的网络都基于 PyTorch 实现。

硬件包括：服务器 1 台，服务器含有 2 块 Intel Xeon Processor E5-2630 CPU，128 GB 内存，32TB 硬盘。

4.3 测试流程

为了对比随机样本投毒和低 P-value 样本的差别，我们选择根据梯度排序后的前 20 个特征，并将这 20 个特征插入 40 个投毒样本中。实验分为重训练和更新两种情况。

基于重训练的投毒具体细节如下：

1. 利用少量样本建立子模型(30000 恶意样本:30000 良性样本)，计算子模型良性样本集合的良性 P-value，得到良性 P-value 最低的 40 个良性样本；
2. 将 20 个特征插入 40 个低 P-value 良性样本中，然后放入训练集并重训练(训练设置与目标模型 1 保持一致)；
3. 训练完成后，将 20 个特征插入恶意样本测试集，并用目标模型 1 对恶意样本重新预测并计算躲避率。

基于三个不同的子模型，重复三次实验。然后我们将低 P-value 良性样本替换为随机样本，同样进行三次实验。对比如图 3.4 所示。

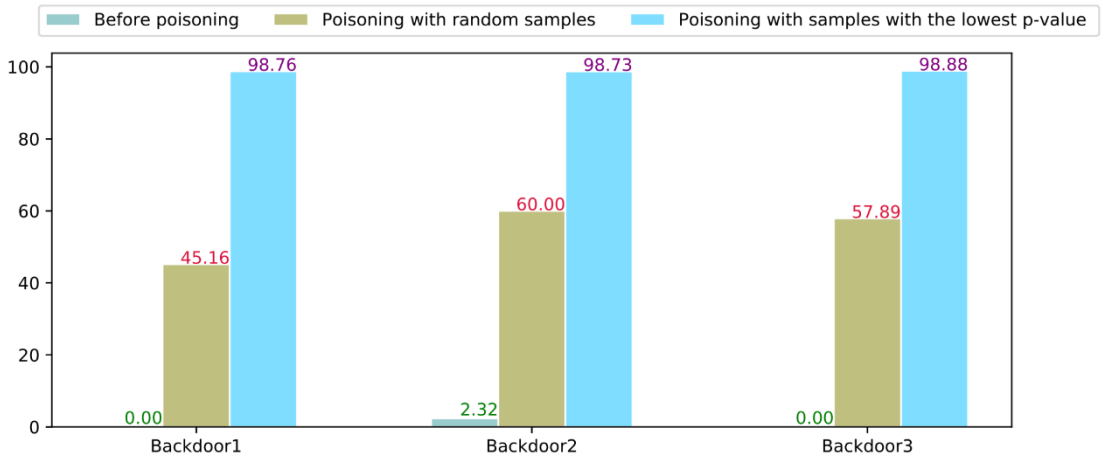

图 4.4 重训练前后躲避率示数变化图

基于模型更新的投毒具体细节如下：

1. 攻击者利用少量样本建立子模型(30000 恶意样本:30000 良性样本)，计算子模型良性样本集合的良性 P-value，得到良性 P-value 最低的 40 个良性样本；
2. 将 20 个特征插入 40 个良性样本中，然后放入更新集并更新(训练设置与目标模型 2 保持一致)；
3. 训练完成后，将 20 个特征插入恶意样本测试集，并用更新后的模型对恶意样本重新预测计算躲避率。

基于三个不同的子模型，重复三次实验。然后我们将低 P-value 良性样本替换为随机样本，同样进行三次实验。模型更新的对比如图 3.5。

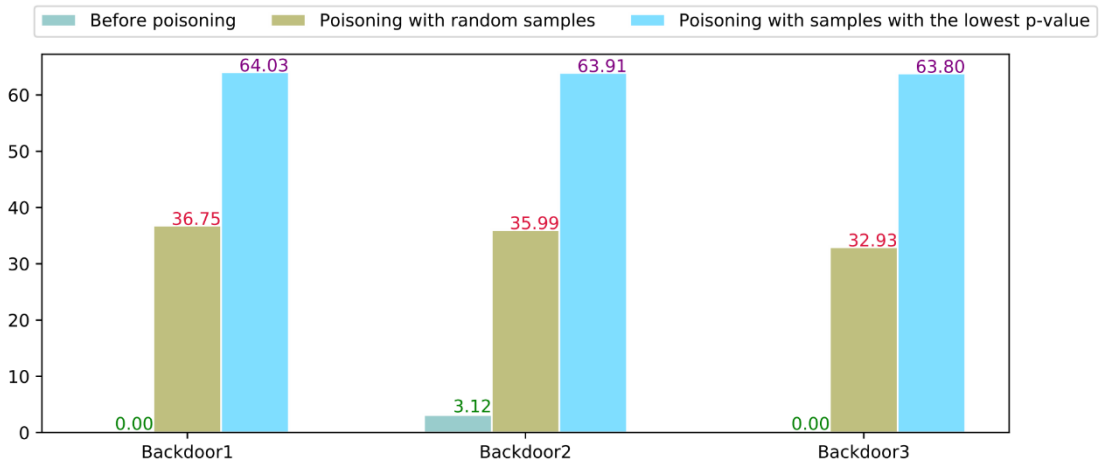

图 4.5 更新前后躲避率示数变化图

4.4 数据测试总结

可见即使不控制样本标签，本文所提出的新攻击策略也可以成功。且无论是更新还是重训练，低 P-value 样本的投毒效果都要明显好于随机样本。

5. 结论

本毒饵攻击策略引入 Conformal Prediction 统计学习算法计算容易影响模型的边缘样本，设置本地对抗模型筛选出信息熵高的关键特征，实现更高效的投毒攻击。该方案对攻击者的要求低、隐蔽性强。通过实验验证，该策略攻击成功率很高，现有的防御系统无法抵抗，是一种高效的投毒攻击技术。

参考文献

[1]. Gu, Tianyu, Brendan Dolan-Gavitt, and Siddharth Garg. "Badnets: Identifying vulnerabilities in the machine learning model supply chain." arXiv preprint arXiv:1708.06733 (2017).

[2]. Liu, Yingqi, et al. "Trojaning attack on neural networks." (2017).

[3]. Liu, Yuntao, Yang Xie, and Ankur Srivastava. "Neural trojans." 2017 IEEE International Conference on Computer Design (ICCD). IEEE, 2017.

[4]. Jordaney, Roberto, et al. "Transcend: Detecting concept drift in malware classification models." 26th {USENIX} Security Symposium ({USENIX} Security 17). 2017.

[5]. Yuan, Zhenlong, et al. "Droid-sec: deep learning in android malware detection." ACM SIGCOMM Computer Communication Review. Vol. 44. No. 4. ACM, 2014.

[6]. Zhu, Hui-Juan, et al. "DroidDet: effective and robust detection of android malware using static analysis along with rotation forest model." Neurocomputing 272 (2018): 638-646.

[7]. Su, Jiawei, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One pixel attack for fooling deep neural networks." IEEE Transactions on Evolutionary Computation (2019).

[8]. Sarkar, Sayantan, et al. "UPSET and ANGRI: Breaking High Performance Image Classifiers." arXiv preprint arXiv:1707.01159 (2017).

[9]. Cisse, Moustapha, et al. "Houdini: Fooling deep structured prediction models." arXiv preprint arXiv:1707.05373 (2017).

[10]. D. Beganovic and E. Smirnov, "Ensemble Cross-Conformal Prediction," 2018 IEEE International Conference on Data Mining Workshops (ICDMW), Singapore, Singapore, 2018, pp. 870-877

[11]. Arp, Daniel, et al. "Drebin: Effective and explainable detection of android malware in your pocket." Ndss. Vol. 14. 2014.

[12]. Gu Jiu-Xiang, Wang Zhen-Hua, Jason Kuen, et al. Recent advances in convolutional neural networks. arXiv: 1512.07108v5, 2017.

针对 Android 恶意代码检测模型投毒攻击方法研究

郭帅[1]，李关玲[1]，胡琪[1]，陈城义[1]，王志[1]

(1. 南开大学 网络空间安全学院 ,天津 300350)

摘　要：采用以机器学习为基础的恶意程序分析已经逐渐取代了对恶意程序的手工分析方式，尽管机器学习算法的使用极大提高了恶意代码检测的发展，但因为其中存在着一定的弱点和问题，因此产生了一些针对于机器学习算法的新型恶意代码，投毒攻击作为其中一种，可以通过在机器学习检测模型中建立后门，诱导训练模型做出错误的分类，从而实现躲避检测模型的检测。本文主要提出了一种基于 *p-value* 的后门注入攻击方法，以 Android 系统常用的 Drebin 算法恶意软件检测系统作为被攻击对象，采用该方法能突破现存的防御，通过观察模型的躲避率和性能,做到在尽量减小因为投毒而造成其他功能损失的前提下，成功将其针对毒样的判定标签改为目标标签，在投毒率小于 0.01%的情况下实现了 99%的躲避率。

关键词：恶意代码检测;投毒攻击;后门注入攻击;Android

中图分类号：TP309.5　　文献标志码：A

Android Malware Poisoning Attacks on Machine Learning based Detection System

GUO Shuai[1],LI Guanling[1],HU Qi[1],CHEN Chengyi[1], WANG Zhi[1]

(1. College of Cyber Science, Nankai University, Tianjin 300350, China)

Abstract: Machine learning based analysis of malicious programs has gradually replaced the manual analysis of malicious programs. Although the use of machine learning algorithm has greatly improved the development of malicious code detection, some new malicious code targeted at machine learning algorithm has been generated due to some weaknesses and problems. As one kind of poison attack, the detection of evading detection model can be realized by building a back door in the machine learning monitoring model and inducing the training model to make wrong classification. This paper mainly presents a targeted poisoning method based on p-value.The Drebin malware detection system, which is based on the SVM algorithm commonly used in the Android system, is used as the target of the attack, and the backdoor injection poison attack algorithm is adopted to make the attack break through the existing defense. By observing the avoidance rate and performance of the model, the SVM was successfully changed from a decision tag for poison samples to a target tag on the premise of minimizing other function losses caused by poisoning. At the same time, we achieved a 99% avoidance rate with a poisoning rate of less than 0.01%.

Key words: Malware Detection; Poison Attack; Backdoor Attack; Android

目前，机器学习被广泛部署在各个安全关键领域，例如金融、医学、司法系统、网络安全或自动驾驶汽车[1]。机器学习算法使用训练集中已经标记过标签的样本自动学习从而形成分类模型，而不是预先确定将输入映射到标签的规则和模型。然后，它可以应用这些模型来预测测试集中新样本的标签。但与此同时，攻击者可以知道机器学习系统的部分或全部参数，甚至直接获得机器学习模型和训练样本集，并利用这些已掌握信息进行训练或测试从而实现攻击。

从现在的研究趋势看，最近的研究主要集中在躲避攻击[1]，这是一种可以导致对特定样本进行定向错误分类的攻击方法，这种攻击的工作原理是对目标样本进行精心修改，使其跨越模型的决策边界，而不改变训练过程或决策边界本身。但这种攻击方法也存在一些不适用的情形，多为一些对手无法控制目标样本的情况。一般在遇到躲避攻击无法完成目的的情况下，就会采用投毒攻击的方法来达到目的[2]，近几年研究[3]也表明了定向投毒攻击的可行性，这种攻击通常将精心设计的实例混合到训练集中，以将模型的边界推向目标。因此，它们为对手无法修改的实例启用了错误分类从而达到了目的。

目前的投毒攻击方法大致可分为两种，第一种为无差别投毒攻击，目的是使恶意代码检测模型发生偏移并遭到破坏，这种攻击方法并不具有针对性，并且隐蔽性也并不理想，目的并非为了使某一个恶意软件绕过检测而是为了使防御者的防御模型不能

正常运转。第二种为针对性投毒攻击，相比于无差别投毒攻击，这种攻击方法是为了能够使自己特指的某个恶意软件能够顺利绕过检测被归为良性，方法是通过将自己精心修饰的样本放入训练集中，在对模型造成尽可能小的不必要的损失的情况下，使得恶意代码检测模型的边界发生一个较小偏移，造成攻击者期望的恶意样本能够最终被归到良性当中。目前针对于该种攻击方式，Radu Marginean 等人提出的stingray攻击算法实现了在确保对机器学习模型性能损失足够小的前提下确保恶意软件被归为良性[1]。除此之外，近几年提出了一种较为新颖的针对性投毒攻击方法——注入后门投毒攻击[5]，其方式是选择一些样本在注入指定特征或修改标签之后成为毒饵样本并将其放回良性样本集，通过训练使得生成的训练模型中生成一个后门，这个后门对于带有指定特征的样本会极大概率检测为良性。

本文提出了一种基于 *p-value* 的后门注入攻击方法，该方法无需修改样本标签便能实现的注入后门投毒攻击。同时，与随机选取样本作为毒样的方法相比，该方法只需注入少量的毒饵样本便可实现较高的躲避率。

1 相关工作

本节将介绍 Android 恶意代码检测系统 Drebin，并介绍投毒攻击的相关研究。

1.1 Drebin

Drebin 是一种用于检测 Android 恶意软件的静态分析方法，可自动推断生成检测模型，并可直接在智能手机上部署，并通过检测模型识别恶意软件，具有迅速高效的优点[6]。

为了检测智能手机上的恶意软件，Drebin 需要全面而轻量级的应用程序表示，以确定恶意活动的典型指示。为此，它采用广泛的静态分析，从不同来源提取特征集并映射到向量空间中分析这些特征集。主要步骤如下：

1) 应用程序的静态分析

首先从每个应用程序的 apk 文件中找到清单文件（AndroidManifest.xml），该文件提供支持安装和以后执行应用程序的数据，使用 Android 打包工具可以在设备上有效地检索存储在此文件中的信息。随后从清单文件中选取硬件组件、请求权限、应用程序组件以及过滤意图四种程序特征[7]。首先是硬件组件，这是所有应用程序所必需的组件。所有硬件，如照相机、蓝牙等，都需要在列表文件中预先声明。特定硬件组件的权限自然存在一定的安全风险，如网络和 GPS 权限，使得恶意应用能够监控用户的位置信息。其次是请求权限：Android 有多种安全机制，权限系统就是其中之一。安装时，它会向用户请求访问相关资源的权限。另一方面，恶意软件比良性应用程序更可能请求某些特定权限。如，当恶意软件需要发送短消息时，它会请求有权发送短消息。而应用程序组件包含四个组件，每个组件定义了系统的不同接口。这些组件的名称收集在功能集中，因为这些名称可能有助于识别已知的恶意软件变体。例如，一些所谓的 droidkungfu 家族的变体共享特定服务的名称。最后是过滤意图，Android 上的进程间和进程内通信主要是通过意图来实现的。作为异步消息交换的被动数据结构，它允许不同的组件和应用程序共享事件信息，而恶意软件经常监听特定意图。恶意软件中涉及的一个典型意图消息示例是引导完成，用于在重启智能手机后直接触发恶意活动。

由于 Android 应用通常被编译为 Dalvik 字节码，而这种字节码易于反汇编。Drebin 实现了一个轻量级反汇编工具，从 dex 文件中选取受限 API 调用、使用权限、可疑 API 调用以及网络地址这四种程序特征[8]。受限 API 调用是指 Android 权限系统限制对一系列关键 API 调用的访问。通过在反汇编代码中搜索这些调用，以深入了解应用程序的功能。暴露恶意行为的一个特定情况是使用不请求所需权限的受限 API 调用。这可能表明恶意软件正在尝试利用根漏洞突破平台限制。使用权限是从受限 API 调用中提取的完整调用集，作为确定请求和实际使用的权限子集的基础。与受限制的 api 调用不同，此功能集提供了应用程序行为的更一般的视图，因为一个权限保护可以保护多个 api 调用。可疑的 API 调用是一些 API 调用允许访问智能手机上的敏感数据或资源，通常在恶意软件示例中找到。由于这些调用特别可能导致恶意行为，因此会将它们提取并收集到单独的功能集中。网络地址是一些恶意应用程序定期建立网络连接以检索从设备收集的命令或越狱数据。因此，反汇编代码中找到的所有 IP 地址、主机名和 URL 都包含在最后一组函数中。其中一些地址可能涉及僵尸网络，因此存在于多个恶意软件样本中，这有助于改进检测模式的学习。

共计八种特征作为分析特征集，如表 1 所示。

表 1　模型特征提取

特征类	Manifest	Code
硬件组件	√	
请求权限	√	
应用程序组件	√	
过滤意图	√	
受限 API 调用		√
使用的权限		√
可疑 API 调用		√
网络地址		√

2) 嵌入向量空间

由于大多数学习方法对数值向量进行操作，我们首先需要将提取的特征集映射到向量空间。为此，我们定义了一个集合 S，它包含 8 个特征集中包含的所有可观察字符串。

$$S := S_1 \cup S_2 \cup \cdots \cup S_8 \tag{1}$$

使用集合 S，我们定义一个 S 维向量空间，其中每个维度为 0 或 1。由映射 φ 将样本映射到该空间，这样对于从 x 提取的每个特征，其所对应的维度为 1，剩余其他维度都为 0。从形式上讲，这个映射 φ 定义如下：

$$\varphi: X \to \{0,1\}^{|s|}, \varphi(x) \to (I(x,s))_{s \in S} \tag{2}$$

其中指标函数 $I(x,s)$ 简单定义如下：

$$I(x,s) = \begin{cases} 1, & \text{安卓应用包含该特征} \\ 0, & \text{安卓应用不包含该特征} \end{cases} \tag{3}$$

最终使每个样本的特征子集构成为一个多维向量，因为一般而言各个特征的存在都极为稀疏，所以一般采用散列表来存储和表示，其向量化过程如图 1 所示。

3) 基于学习模型的检测

随后运用线性 SVM 来进行训练。将给定恶意类和良性类的向量作为训练数据，线性 SVM 确定超平面：

$$W^T X + b = 0 \tag{4}$$

该超平面用最大余量分隔两类，生成检测模型。

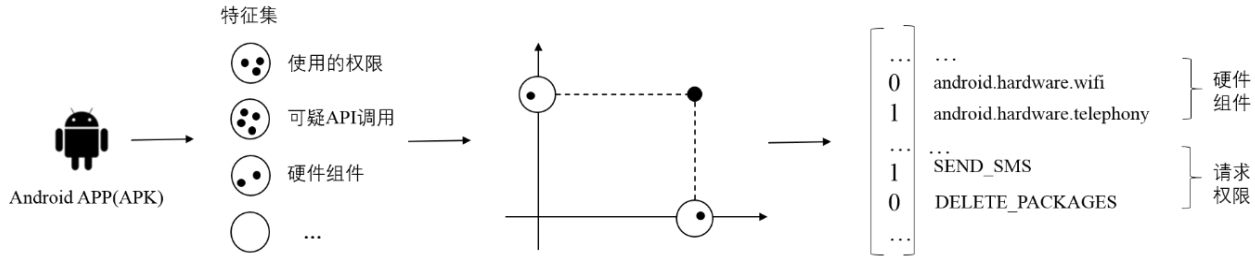

图 1 向量化过程

1.2 投毒攻击

在机器学习领域，通常有两种类型的攻击:投毒攻击和躲避攻击。一般来说，躲避攻击是根据已生成的恶意代码检测模型中各特征所起到的作用和权重来改变目标恶意样本的特征，使恶意样本逐渐绕过边界[9]。而投毒攻击又分为针对性攻击和无差别攻击。其中，无差别攻击的目的是破坏模型的可用性，它并没有针对于某一恶意样本进行投毒致使模型只对该恶意样本做出错误判断，而针对性投毒攻击的目的是通过投毒使某一样本或某一种样本逃避模型的检测，具有隐蔽性和实用性[1]。

无差别攻击，其目的是破坏模型的可用性，与针对性投毒攻击的区别在于，它并没有针对于某一恶意样本进行投毒致使模型只对该恶意样本做出错误判断，而是进行一个无目的的投毒使得模型对于很多恶意样本做出错误的判断[10]。

针对性投毒攻击更加具有针对性和定向性，基本原理是将目标恶意样本中存在的特征取出，并将其插入到良性样本训练集当中，并进行训练，这样生成的训练模型会将带有这些特征的样本极大概率识别为良性，从而实现定向的攻击，如图 2 所示。

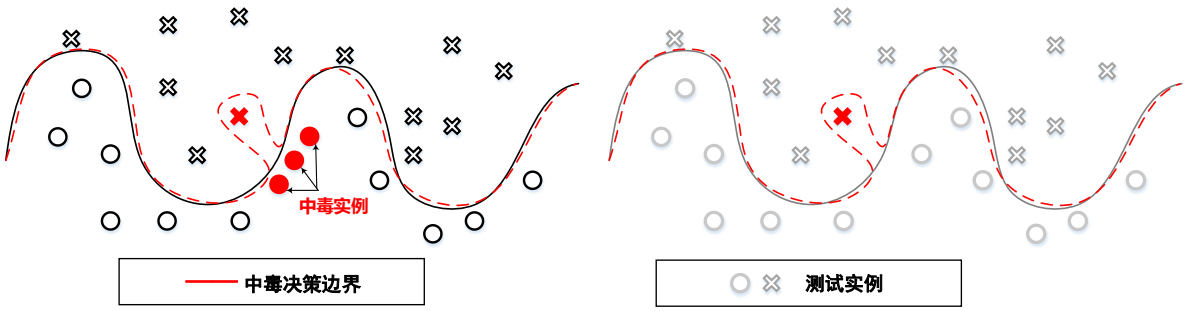

图 2 针对性投毒攻击

目前产生了一种全新的投毒攻击方式——注入后门攻击[11]，它的工作原理是在正常模型流程的基础上，在良性样本训练集当中添加若干指定特征，然后放入机器学习算法当中供其训练，最后训练模型，这个模型会大概率将含有这些特征的测试样本识别为良性，随后在目标样本中添加入这些特征，这样的话该样本就能通过后门绕过训练模型的检测从而被识别为良性，如图 3 所示。

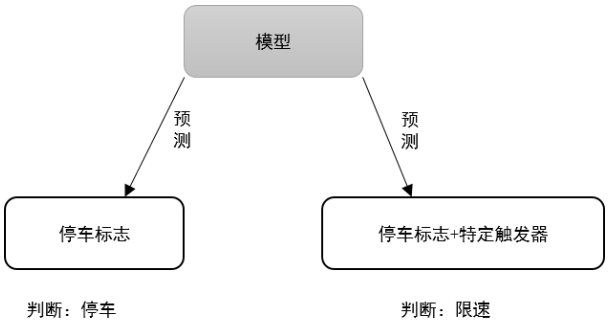

图 3 后门注入攻击

2 基于 *p-value* 的后门注入攻击

本文主要目的是实现一种无需对样本标签进行修改便可实现的注入后门投毒攻击方法。除此之外，该方法须具有较好的针对性，能够实现较高的躲避模型检测的概率，并且尽量减少因投毒而造成的对检测模型的性能损失，从而有利于提高投毒的隐蔽性而减小被发现的概率。

2.1 后门攻击分析

机器学习模型都是通过基于已有数据训练来学习底层分布的，对于边界附近或分类错误的样本，为了正确预测，对模型权重影响较大。而对于远离边界或能很好分类的样本，不需要对模型进行调整就能很好的分类。

攻击者为了获得一个特定的后门，需要让投毒样本离原始类簇足够远，在这种情况下，模型会对后门特征的权重进行较大改动。而其他特征权重由于在数据底层分布较为稳定，对其进行改变会破坏该模型的性能。则模型会主要对后门特征做出较大调整来保证投毒样本的正确分类，而同时维护其他特征的权重尽可能不变。

本文用一个简单实例来证明这一点，如下：

1) 随机生成一些三维数据，但第三维全为 0。

2) 建立模型如图 4 所示，模型权重为[2.44,-0.91,0.31],偏差为-8.3。

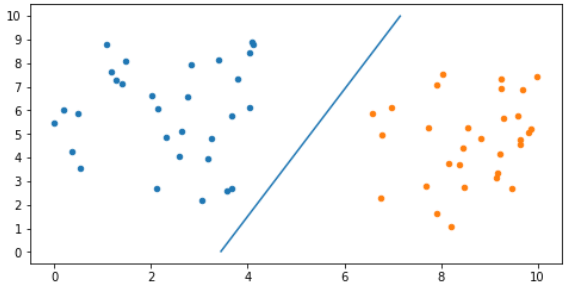

图 4 简单逻辑回归模型

3) 插入一个离类簇远的样本 x2，然后重训练如图 5。模型的权重是[2.2833, -0.6479, -4.0875]，偏差是-8.9036。

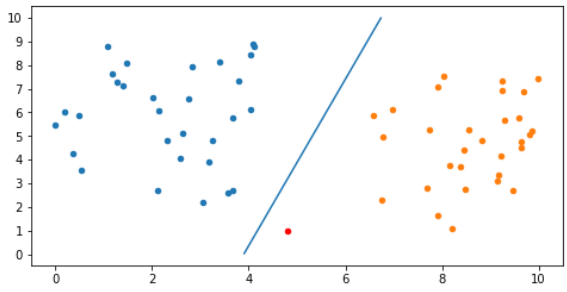

图 5 被投毒后的模型

由图 5 可见，当我们选择一个距离目标类簇较远的样本时，模型在前两维特征的维度下，几乎没有改变。但第三维特征已经下降到了一个非常低的值。从而如果某个样本存在第三维的特征则其分类结果便会受到很大的影响。

2.1 方法设计

依照前文当中对于注入后门攻击的介绍，我们需要从良性训练集中挑选一定数量的样本，随后在其中注入指定特征生成毒饵样本，将其放入 Drebin 算法中训练，最终获得一个注入后门的模型，然后将恶意测试集中的目标样本注入指定特征，通过观察躲避率，从而分析该后门攻击的效果。除此之外，因为注入后门的投毒攻击具有相对隐蔽不易被发现的特性，为了验证所注入毒饵样本对模型的性能影响并不明显，我们需要将已经生成好的模型的性能同正常模型的性能进行分析，进而得到其性能的损失大小，若性能变化不大，则说明该方法隐蔽性较为良好。整体方法流程如图 6 所示。

以下将对实验的步骤以及设计方案进行介绍：

1) 计算得分

从良性样本集中选取特定的若干样本，在选取的时候，我们应该选取 *p-value* 尽量低的良性样本，因为在对应到 Drebin 算法的良性样本集中，*p-value* 越低的良性样本所对应的特征向量在坐标空间当中对应的坐标越靠近于超平面的边际，这种样本在注入指定特征后能够更好地使超平面的边际产生倾斜，从而更好地将注入后门的恶意样本检测为良性从而绕过检测[12]。

为了获得良性样本的 *p-value*，我们首先需要获得到每个样本在样本集中的得分，这个得分对应到 SVM 中可以具体为该样本的向量坐标到超平面的距离，距离超平面的距离越远，说明其在所有样本集中的可信程度越高。

为了获得训练集的得分，我们可以采用 k-fold 交叉验证的方法从中选取一部分作为训练集，另一部分作为测试集，得到测试集距离超平面的距离即为得分。例如在本次试验中，我们将训练集分为 50 等份，每次训练时从中挑取一份作为测试集，其余 49 份作为训练集，生成训练模型之后用来获得测试集的得分，这样循环操作 50 次，最终获得全体训练集的得分。

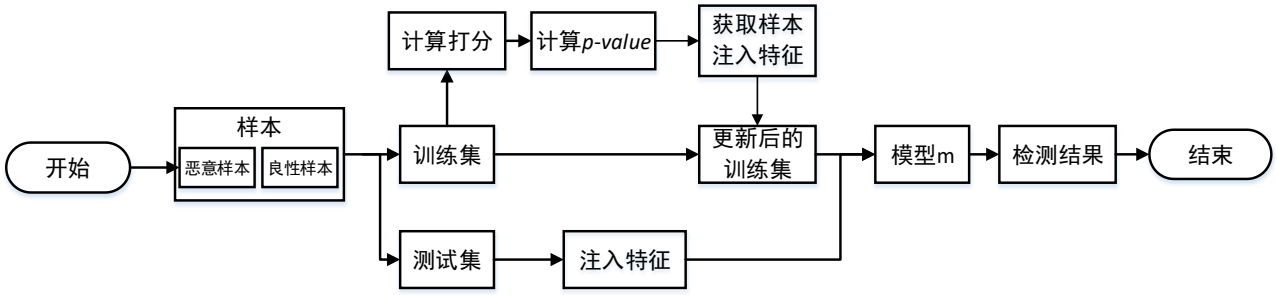

图 6 基于 p-value 的定向投毒方法流程图

在交叉验证完成后，会得到每个样本的得分以及被模型预测的标签。

2) 计算 *p-value*

在获得得分之后，我们根据 *p-value* 的计算公式，要得到每一个测试数据的得分在整个训练集得分中的不一致性程度，*p-value* 越大，其可信程度越高[13]。我们需要计算 *p0* 和 *p1* 这两个 *p-value*，其中 *p0* 表示预测标签为良性的 *p-value* 的值，*p1* 表示预测标签为恶意的 *p-value* 值。据此，我们可以得出该实验中 *p-value* 的公式如下：

$$p-value=\frac{\text{训练集中比该样本得分 小的样本个数}}{\text{样本总数}} \quad (5)$$

3) 选择样本作为毒饵样本

在获得 *p-value* 之后，我们需要获得的样本是所有真实标签为良性当中 *p0* 最小的样本，从中挑选一定数量的样本为后续产生毒饵样本集做准备。

4) 注入指定特征

在选取好样本集和特征之后，将特征注入到样本集中，构成毒饵样本集。

5) 训练模型

将毒饵样本集放入训练集当中，采用 Drebin 进行训练得到训练模型。将恶意测试集中的目标样本也注入选好的指定后门特征，并将其放入训练模型中进行测试，观察目标恶意样本是否被检测为良性，同时观察所生成的模型的性能损耗。

3 实验

本章主要基于对前面所介绍的投毒攻击的方法为依据，以 Drebin 为被攻击算法，实现了投毒攻击，同时也介绍了实验数据的来源和分组，并针对实验结果给出详细分析。

3.1 实验数据来源

本次所用的样本来源于 AndroZoo，它从多个 Android 应用市场收集了大量的 Android 应用程序，例如 Android 官方的 Google Play 应用程序市场等。AndroZoo 的样本库目前包含 8 939 767 个不同的 APK 样本，每个 APK 样本都在 VirusTotal 平台上面进行了检测，并根据检测结果对样本打标签。VirusTotal 平台上面集成了几十个杀毒软件公司开发的杀毒引擎。Drebin 是在 2014 年提出，为了在实验中体现 Drebin 的最佳表现，我们从 AndroZoo 中选取了 2014 年 1 月至 2015 年 12 月期间的应用程序数据进行实验，其中包含 APK 样本 200 000 个，共提取特征超过 166 万。APK 样本被分为训练集和测试集，其中训练集有 18 万 APK 样本，测试集中有 2 万 APK 样本，具体数字如表 2 所示。

表 2　实验样本分布

	训练集	测试集	共计
恶意样本	90 000	10 000	100 000
良性样本	90 000	10 000	100 000
共计	180 000	20 000	200 000

3.2 实验评估体系

为了方便分析实验结果，对实验效果做出量化分析，我们引入机器学习算法和投毒攻击

中几个常用的衡量指标，便于对实验结果的对比分析。

在本次实验，我们选择了机器学习算法常用的精确率（*precision*）、召回率（*recall*）和综合评价指标（*f*1-*score*）作为衡量检测模型性能的评价指标[14]，同时引入躲避率（*evade*）衡量投毒攻击的效果。精确率体现的是预测正确的数据占全部数据的比值。召回率体现的是所有预测为恶意 APK 的样本中，预测正确的比率。综合评价指标 *f*1-*score* 则是综合考虑精确率、召回率后给出的计算结果，能够综合地代表和体现模型性能。躲避率代表了所有恶意样本中被预测为良性样本的概率，躲避率越高，表示投毒攻击的效果越好。

具体评价参数和评价指标公式如表 3、4 所示。

表 3　评价参数分布

评价参数	解释
TP（True Positive）	预测答案正确
FP（False Positive）	将其他类别预测为本类
FN（False Negative）	本类标签预测为其他类标签

表 4　评价指标公式

评价参数	解释
精确率 / 查准率(*precision*)	$precision_k = \frac{TP}{TP + FP}$
召回率 / 查全率 (*recall*)	$recall_k = \frac{TP}{TP + FN}$
准确率(*accuracy*)	$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$
F1 分数（*f1-score*）	$f1_k = \frac{2 \cdot precision_k \cdot recall_k}{precision_k + recall_k}$
躲避率（*evade*）	$evade = \frac{FN_{malware}}{TP_{malware} + FN_{malware}}$

3.3 实验分组

除了前文提到的未被投毒攻击的检测模型和被投毒之后的检测模型之外，为了能够更好地进行投毒效果的对比和分析，我们还构建了第三个检测模型。第三个模型也受到了投毒攻击，但是攻击使用的毒饵样本不是基于 *p-value* 构建的，而是随机选取了良性样本来构建的毒饵样本，毒饵样本的数量与使用 *p-value* 构建的毒饵样本数量相同。因此，总共基于三个 Android 恶意代码检测模型进行了对比实验，分别是正常情况下不受攻击的检测模型（m1）、选取训练集中 x 个 *p-value* 低的良性样本构造毒饵样本后投毒的检测模型（m2）和随机选取的 x 个良性样本来构建毒饵样本后投毒的检测模型（m3），检测结果的评价指标有躲避率、精确率、召回率和 *f*1-*score*。

除此之外，为了更好比较出毒饵样本数量多少对攻击效果的影响，分别选取了 x 个毒饵样本进行实验，其中 x=1、5、10、15、20、25、30、35、40，每次实验都重新生成检测模型 m2 和 m3，并对比分析 m2 和 m3 的检测结果。

3.4 实验结果

通过对检测模型进行投毒攻击，在不断改变毒饵样本数量后，三个检测模型躲避率的结果如图 7 所示。

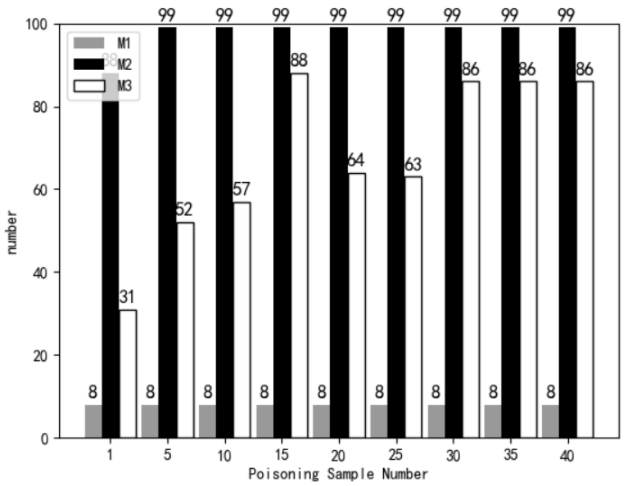

图 7 躲避率实验结果

通过实验结果数据可以看出，相比于随机选取毒饵样本的攻击，选取 *p-value* 低的样本构造毒饵样本能够产生更高的躲避率。基于随机样本的投毒攻击在原有训练集基础上添加少量毒饵样本，几乎不会影响整个样本空间的分布规律。

而对于低 *p-value* 值的 APK 样本，它们一般都对模型的决策边界起到了决定性的作用，因此投毒攻击的效果要优于普通的 APK 样本。

选取 *p-value* 值低的样本构造毒饵样本，攻击后的躲避率很高。当毒饵样本的数量达到 5 个以后，躲避率就已经达到 99%，说明注入后门方法只需要很少的毒饵样本就能够获得很高的躲避率。

为了验证投毒攻击的隐蔽性，实验中收集了 m1 检测模型的精确率、召回率和 *f*1-*score*，其实验结果如表 4 所示。

表 5　m1 模型性能

	精确率	召回率	*f*1-*score*
Malware	0.91	0.92	0.92
Goodware	0.92	0.91	0.91
Total	0.91	0.91	0.91

随后对检测模型 m2 和 m3，在毒饵样本数量分别为 1、5、10、15、20、25、30、35、40 的情况下的精确率、召回率和 *f*1-*score* 进行记录，发现实验结果均与检测模型 m1 的结果基本一致，由此可以证明后门攻击具有很好的隐蔽性，没有导致被攻击模型的检测精确率、召回率和 *f*1-*score* 值的下降。

4 结论

Android 应用程序的安全问题日益受到关注，基于机器学习算法的 Android 恶意代码检测模型的应用越来越广泛，模型自身的安全问题逐渐成为安全领域的研究热点。

本文在阅读了近些年来相关领域文献的基础上，基于统计学习算法 p-value 值，提出了一种新的投毒攻击方法，成功实现了针对于 Drebin 模型的投毒攻击。实验结果显示，基于 p-value 值的投毒攻击，通过构造小于 0.01%的毒饵样本，就可以在 Drebin 检测模型上创建一个后门。该后门不影响 Drebin 模型检测结果的准确率和召回率，具有隐蔽性。对于知道后门的攻击者，通过后门特征的嵌入，可以使 Android 恶意代码达到 99%的躲避率。

参考文献 (References)

[1] SUCIU O, MARGINEAN R, KAYA Y, et al. When does machine learning FAIL? Generalized transferability for evasion and poisoning attacks [C]// 27th USENIX Security Symposium (USENIX Security 18). Baltimore, USA, 2018: 1299-1316.

[2] PAPERNOT N, MCDANIEL P, SINHA A, et al. Towards the science of security and privacy in machine learning [J]. CoRR abs, 2016,1611: 03814.

[3] BARRENO M, NELSON B, JOSEPH A D, et al. The security of machine learning [J]. Machine Learning, 2010, 81(2): 121-148.

[4] KURAKIN A, GOODFELLOW I, BENGIO S. Adversarial machine learning at scale [J]. CoRR abs, 2016, 1611: 01236.

[5] LIAO C, ZHONG H, SQUICCIARINI A, et al. Backdoor embedding in convolutional neural network models via invisible perturbation[J]. CoRR abs, 2018, 1808: 10307.

[6] ARP D, SPREITZENBARTH M, HUBNER M, et al. Drebin: effective and explainable detection of android malware in your pocket [C]// Ndss. California, USA, 2014: 14: 23-26.

[7] BIGGIO B, NELSON B, LASKOV P. Poisoning attacks against support vector machines [C]// ICML. Scotland, UK, 2012.

[8] YUAN Z L, LU Y Q, WANG Z, et al. Droid-sec: deep learning in android malware detection [C]// ACM SIGCOMM Computer Communication Review. Chicago, IL, USA, 2014: 44(4): 371-372.

[9] XU L, ZHAN Z, XU S, et al. An evasion and counter-evasion study in malicious websites detection [C]// 2014 IEEE Conference on Communications and Network Security. San Francisco, CA, USA, 2014: 265-273.

[10] YANG C F, WU Q, LI H, et al. Generative poisoning attack method against neural networks [J]. CoRR abs, 2017, 1703: 01340.

[11] GU T Y, DOLAN-GAVITT B, GARG S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. CoRR abs, 2017, 1708: 06733.

[12] JORDANEY R, SHARAD K, DASH S, et al. Transcend: Detecting concept drift in malware classification models [C]// 26th USENIX Security Symposium (USENIX Security 17). Vancouver, BC, Canada, 2017: 625-642.

[13] NOURETDINOV I. Validity and efficiency of conformal anomaly detection on big distributed data [J]. Advances in Science, Technology and Engineering Systems Journal, 2017: 254-276.

[14] JI Y Z, ZHANG X Y, WANG T. Backdoor attacks against learning systems [C]// 2017 IEEE Conference on Communications and Network Security (CNS). Las Vegas, NV, USA, 2017: 1-9.