软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(4):1926−1943 [doi: 10.13328/j.cnki.jos.006722]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于模型解释的 PE 文件对抗性恶意代码检测*

田志成[1], 张伟哲[1,2], 乔延臣[2], 刘 洋[1,2]

[1](哈尔滨工业大学(深圳) 计算机科学与技术学院, 广东 深圳 518055)

[2](鹏城实验室, 广东 深圳 518055)

通信作者: 张伟哲, E-mail: wzzhang@hit.edu.cn

摘 要: 深度学习已经逐渐应用于恶意代码检测并取得了不错的效果. 然而, 最近的研究表明: 深度学习模型自身存在不安全因素, 容易遭受对抗样本攻击. 在不改变恶意代码原有功能的前提下, 攻击者通过对恶意代码做少量修改, 可以误导恶意代码检测器做出错误的决策, 造成恶意代码的漏报. 为防御对抗样本攻击, 已有的研究工作中最常用的方法是对抗训练. 然而对抗训练方法需要生成大量对抗样本加入训练集中重新训练模型, 效率较低, 并且防御效果受限于训练中所使用的对抗样本生成方法. 为此, 提出一种 PE 文件格式恶意代码对抗样本检测方法, 针对在程序功能无关区域添加修改的一类对抗样本攻击, 利用模型解释技术提取端到端恶意代码检测模型的决策依据作为特征, 进而通过异常检测方法准确识别对抗样本. 该方法作为恶意代码检测模型的附加模块, 不需要对原有模型做修改, 相较于对抗训练等其他防御方法效率更高, 且具有更强的泛化能力, 能够防御多种对抗样本攻击. 在真实的恶意代码数据集上进行了实验, 实验结果表明, 该方法能够有效防御针对端到端 PE 文件恶意代码检测模型的对抗样本攻击.

关键词: 对抗样本; 恶意代码检测; 模型解释; 异常检测; 深度学习

中图法分类号: TP309

中文引用格式: 田志成, 张伟哲, 乔延臣, 刘洋. 基于模型解释的 PE 文件对抗性恶意代码检测. 软件学报, 2023, 34(4): 1926−1943. http://www.jos.org.cn/1000-9825/6722.htm

英文引用格式: Tian ZC, Zhang WZ, Qiao YC, Liu Y. Detection of Adversarial PE File Malware via Model Interpretation. Ruan Jian Xue Bao/Journal of Software, 2023, 34(4): 1926−1943 (in Chinese). http://www.jos.org.cn/1000-9825/6722.htm

Detection of Adversarial PE File Malware via Model Interpretation

TIAN Zhi-Cheng[1], ZHANG Wei-Zhe[1,2], QIAO Yan-Chen[2], LIU Yang[1,2]

[1](College of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China)

[2](Pengcheng Laboratory, Shenzhen 518055, China)

Abstract: Deep learning has been used in the field of malware detection and achieved great results. However, recent research shows that deep learning models are not safe, and they are vulnerable to adversarial attacks. Attackers can make malware detectors give wrong output by making a few modifications to the malware without changing the original function, resulting in the omission of malware. To defend adversarial examples, the most commonly used method in previous work is adversarial training. Adversarial training requires generating a large number of adversarial examples to retrain the model, which is inefficient. Besides, the defense effect is limited by the adversarial example generation method used in training. As such, a new method is proposed to detect adversarial malware in PE format, aiming at the type of adversarial attacks that add modification to the function independent area of PE file. By using model interpretation techniques, the decision-making basis of the end-to-end malware detection model can be analyzed and the features of adversarial examples are extracted. Anomaly detection techniques are further used to identify adversarial examples. As an add-on module of the malware detection model, the proposed method does not require modifying the original model and does not need to retrain the model. Compared with other defense

* 基金项目: 广东省重点领域研发计划(2019B010136001); 深圳市基础研究专项资金(JCYJ20190806143418198)
收稿时间: 2020-07-04; 修改时间: 2021-03-18, 2021-07-30; 采用时间: 2022-06-07

methods such as adversarial training, this method is more efficient and has better generalization ability which means it can defend against a variety of adversarial attack methods The proposed method is evaluated on a real-world dataset of malware. Promising results show that the method can effectively defend the adversarial attacks against the end-to-end PE format malware detection model.
Key words: adversarial examples; malware detection; model interpretation; anomaly detection; deep learning

随着深度学习被广泛应用于图像识别[1]、语音识别[2]、自然语言处理[3]等领域, 将深度学习技术应用于网络空间安全领域的研究也逐步涌现[4,5], 其中最重要的应用场景之一就是恶意代码检测[6]. 传统的基于签名的恶意代码检测方法通过构造规则识别有特定特征的恶意代码, 但构造规则的方法泛化能力弱, 对于新出现的恶意代码识别效果较差[7]. 因此, 研究者及厂商开始关注基于 *n*-gram[8]、基于程序行为特征学习[9]等机器学习方法的恶意代码检测模型. 除此之外, 卷积神经网络、循环神经网络等深度学习技术也被用于恶意代码检测[10−12]. 得益于强大的抽象能力, 深度学习模型可以自动提取恶意代码二进制文件中的特征并进行分类. 此类方法被称为端到端(end-to-end)的恶意代码检测方法[11], 其优势在于训练过程简单, 不需要复杂的特征提取工作以及专业的背景知识, 并且具备防御新威胁的能力[12].

深度学习技术简化了恶意代码检测器的构建过程, 却引入了新的问题: 深度学习方法具有很强的抽象能力, 但其学习和决策过程往往是难以理解的, 这也导致了仅依靠有限数据训练的深度学习模型存在漏洞, 容易遭受攻击[13]. 一些研究证实: 攻击者通过对模型的输入做微量修改, 可以误导深度学习模型, 导致其做出错误的决策, 这类攻击被称为对抗样本攻击[14]. 在计算机视觉领域, 对抗样本攻击已经有了很多成熟的研究[15,16], 在不影响人类理解的前提下向像素值中添加扰动, 构造的对抗样本能够轻松地欺骗训练好的分类器, 使其以高置信度给出错误结果.

与图像对抗样本不同, 恶意代码对抗样本的生成分为特征空间和问题空间两类[17]: 特征空间下, 恶意代码对抗样本仅对样本提取的特征进行修改, 不具备真实的威胁; 而问题空间下的恶意代码对抗样本的生成有较多的限制, 攻击者在构造对抗样本时, 需要保证恶意代码文件的原有功能不被修改[18]. 尽管如此, 精心设计的恶意代码对抗样本仍能够轻松地误导检测器, 使检测器将其误分类为良性软件, 从而成功逃过恶意代码检测器的识别, 实施恶意行为[19].

针对图像对抗样本的防御, 已有的研究工作提出了基于编码和去噪等技术的防御方法[20,21], 但由于恶意代码对抗样本的特征空间和问题空间一般没有直接的映射关系[17], 这些方法无法直接用于恶意代码对抗样本的防御. 针对恶意代码对抗样本防御这一特定问题, 目前已知的研究工作相对较少. 其中: 对于 Android 恶意代码对抗样本, 文献[22,23]提出可以通过对抗训练方法进行防御; 对于 PE(portable executable)文件恶意代码对抗样本的防御, 对抗训练也是现有工作中较为有效的方法[6]. 对抗训练的主要思路是: 将对抗样本加入训练集中重新训练模型,来提高模型的健壮性. 然而, 对抗训练需要较大的时间成本, 且防御的有效范围受训练中使用的对抗样本生成方法的限制, 在实际应用中存在局限性[24,25]. 因此, 针对 PE 文件恶意代码对抗样本问题, 研究一种能够有效防御多种对抗样本攻击的防御方法, 已经成为迫切需求.

本文以 PE 文件格式的恶意代码对抗样本的防御问题为研究对象, 总结分析了现有的针对 PE 文件格式恶意代码的端到端检测模型的对抗样本攻击方法. 通过研究其共有特性, 提出了一种基于模型解释的恶意代码对抗样本防御方法. 为了保证恶意代码原有的恶意功能不被修改, PE 文件恶意代码对抗样本在构造时, 一般在文件中不影响文件功能的区域添加扰动. 由于端到端模型不经过预处理过程, 我们可以利用模型解释方法对模型的决策依据与输入样本的空间位置建立映射, 并基于此分析得到对抗样本中对模型决策有重要影响的字节集中在文件头、文件尾等区域, 这与正常良性样本存在差异. 基于这一发现, 本文通过对恶意代码检测模型决策过程的分析, 提取被判定为良性的输入样本的贡献度分布特征, 进而利用异常检测方法识别恶意代码对抗样本.

本文的主要贡献在于:

(1) 提出了一种基于模型解释的 PE 文件恶意代码对抗样本检测方法, 相较于对抗训练等防御方法, 本方法的优势在于能防御多种针对端到端恶意代码检测模型的对抗样本攻击, 不需要重新训练模型,

能够高效部署, 实验中的防御效果显著优于对抗训练;

(2) 基于模型解释技术, 提出了一种恶意代码检测模型的决策依据分析方法, 利用此方法, 可对模型可信度及鲁棒性进行评估, 帮助理解模型;

(3) 总结了现有的针对端到端恶意代码检测模型的对抗样本攻击方法, 研究了这些方法的异同, 对比了攻击能力, 并针对其局限性研究了对应的防御方法.

本文第 1 节是相关工作的介绍. 第 2 节对端到端恶意代码检测模型及恶意代码对抗样本攻击进行介绍.第 3 节对本文提出的恶意代码对抗样本检测方法进行详细描述. 第 4 节为实验设置与实验结果的分析讨论. 第 5 节为本文的总结.

1 相关工作

深度学习技术被广泛应用于图像识别、自然语言处理等领域的同时, 在 PE 文件恶意代码检测任务上也开始出现基于深度神经网络的研究. Saxe 等人[26]提取了 PE 文件字节熵的直方图作为特征训练了深度神经网络, 在实验中取得了 95%的准确率. Hardy 等人[27]提出了一种 API 调用序列为特征的基于深度学习的恶意代码检测模型, 并在处理特征时使用了栈式自编码器. 此外, 文献[28,29]等工作提出了将 PE 文件的二进制字节、操作码等特征通过可视化转化为图像, 并进一步利用基于深度神经网络的图像分类模型进行分析. Raff 等人[10]提出的 MalConv 恶意代码检测模型第 1 次将 PE 文件的原始字节数据作为输入, 训练端到端的深度神经网络模型. 与前述工作的不同之处在于, 这种方法不需要手动提取特征和预处理. Raff 等人认为, 这样可以最大程度地保留 PE 文件包含的信息, 同时在模型的构建过程中不需要专业的背景知识. 上述的恶意代码检测方法都是静态的, 即检测时不需要运行程序以获取特征. 在文献[30,31]等工作中, 深度学习模型也被用于恶意代码的动态检测, 需要构建仿真环境对样本进行运行时分析, 实际应用的要求较高.

对抗样本攻击的概念最早由 Szegedy 等人[13]提出, 他们还提出了 L-BFGS 对抗样本生成方法,用于生成图像对抗样本. 该方法将生成对抗样本的问题定义为受约束的优化问题, 优化目标设置为在使对抗样本与原始图像距离误差尽量小的前提下, 使分类器将对抗样本判定为特定的类别. Goodfellow 等人[14]对对抗样本的原理进行了解释, 证明了对抗样本的存在是深度网络的高维线性造成的, 并提出了基于梯度下降原理的对抗样本生成方法 FGSM. FGSM 方法通过在梯度方向上对原像素添加增量, 使模型对生成的对抗样本误分类. 上述两个工作为对抗样本生成方法奠定了基础, 后续的衍生方法多数加入迭代机制[15,16], 生成的图像对抗样本具有更好的鲁棒性和对抗性[32].

与图像对抗样本不同, 对于问题空间下的恶意代码对抗样本攻击, 需要保持恶意代码原有的程序功能[17], 因此需要在不影响文件功能的前提下添加扰动. 文献[17,22,23,33]等工作研究了 Android 恶意软件的对抗样本攻击问题, 其中, Pierazzi 等人[17]对恶意软件中问题空间下的对抗样本攻击问题进行了总结, 给出了一种对问题空间下对抗样本问题的形式化定义, 并基于此提出了一种新的 Android 恶意软件对抗样本攻击方法. 针对 PE 文件恶意代码对抗样本的生成问题, Kolosnjaji 等人[18]提出了一种在恶意代码文件尾部添加扰动以生成对抗样本的方法, 在不超过恶意代码检测模型输入长度限制的前提下, 可以添加较多的扰动字节. Kreuk 等人[19]提出了一种在 PE 文件中的节间区域添加扰动的方法, 添加的扰动基于前文提到的 FGSM 方法构造, 获得了较高的成功率. Demetrio 等人[34]通过对恶意代码检测模型的分析, 得出 PE 文件 DOS 头所表现出的特征对模型决策的影响较大, 并提出了一种通过修改恶意代码 DOS 头中不影响文件功能的几十个字节来生成对抗样本的方法, 在少量的扰动下, 可以误导模型的判断. 上述的 3 种恶意代码对抗样本攻击方法都是白盒攻击, 白盒攻击中, 攻击者可以获取模型的相关信息. Hu 等人[35]提出了一种黑盒攻击方法 MalGAN. MalGAN 基于 GAN 方法, 利用原始样本和黑盒模型所产生的输出, 在线下训练一个判别器, 再用判别器指导生成器来产生对抗样本. MalGAN 需要向样本中添加较多的扰动, 效率低于白盒攻击方法. Ishai 等人[36]提出了一种针对基于 API 调用序列的 PE 文件恶意代码检测模型的对抗攻击方法, 利用了 RNN 模型的可迁移性, 通过插入 API 调用实施黑盒攻击.

针对对抗样本的防御问题, 已经有较多的研究工作, 防御思路可以分为对抗性蒸馏[37]、对抗训练[14,38]、预处理[20,39]、正则化[21]等几个类别, 大部分方法主要适用于计算机视觉任务中的对抗样本防御. 其中: 对抗防御方法能够用于对问题空间下恶意代码对抗样本的防御, 并可以获得不错的效果[22]. 然而, 对于端到端的 PE 文件恶意代码检测模型的对抗样本问题, 使用对抗训练进行防御存在两方面的局限性.

- 一是生成扰动的局限. 生成对抗样本的过程一般被归纳为空间搜索的优化问题[13], 而这本质是非凸优化问题, 对抗训练中使用的对抗样本生成方法, 如 FGSM[14], 一般是通过设计损失函数进行近似, 因此训练后的模型会使生成对抗样本更加困难, 但仍存在被攻击的可能[25];
- 二是加入扰动方式的局限. PE 文件恶意代码对抗攻击一般通过在 PE 文件中功能无关的区域插入扰动生成对抗样本, 本文研究了已知的 3 种[18,19,34], 然而真实场景下, 防御方一般无法得知攻击者的意图, 对于不同的扰动位置, 如果训练过程中未知, 就无法生成对应的对抗样本进行对抗训练.

此外, 针对 PE 文件恶意代码抗样本防御问题, Wang 等人[40]提出了一种将待判定样本中的某些特征进行随机化失效的方法, 以削弱对抗样本对模型的误导能力. 但这种方法只适用于基于特征的恶意代码检测模型, 不能用于针对端到端恶意代码检测模型的对抗样本攻击的防御. 值得说明的是: Liu 等人[41]提出的对抗样本防御方法也利用了模型解释思想, 但该方法的核心思想是利用模型解释方法以更有效地生成对抗样本, 再将对抗样本用于对抗训练以提升模型的鲁棒性. 而在本文中, 模型解释方法被用于分析对抗样本与正常样本的差异, 从而识别对抗样本, 进而提升模型的鲁棒性.

针对现有方法的不足, 本文提出了一种基于模型解释的 PE 文件恶意代码对抗样本防御方法, 可以有效防御针对端到端恶意代码检测模型的多种对抗样本攻击, 检测器构建过程中, 通过良性样本训练异常检测模型, 不需要获取攻击样本的特征; 同时, 该方法作为独立的检测模块, 不需要修改或重新训练原模型, 在应用中能够更高效地部署.

2 恶意代码对抗样本攻击

本节首先介绍一种典型的基于深度神经网络模型的恶意代码检测模型, 然后介绍针对此类恶意代码检测模型的对抗样本攻击方法.

2.1 端到端恶意代码检测模型

基于深度学习的端到端恶意代码检测模型以二进制文件为输入, 以恶意或良性的检测结果为输出, 不需要进行特征提取工作. 本文以基于卷积神经网络的恶意代码检测模型为例, 研究针对端到端恶意代码检测模型的对抗样本攻击. 具体地, 我们以 Raff 等人提出的 MalConv[10]恶意代码检测模型为例进行分析, 研究恶意代码检测模型的对抗样本防御方法.

对于端到端恶意代码检测模型, 输入模型的二进制 PE 文件可以看作一个由字节元素组成的长序列, 每一个字节的可取值为 $B=\{0,\ldots,255\}$, 即对于一个 k 字节的文件, 可以表示为$(b_1,\ldots,b_k)$, 并且$(b_1,\ldots,b_k)\in B^k$. 由于检测模型的输入需要是固定长度 d 的向量, 对于 $k<d$ 的文件, 通过在文件末尾补 0 构造输入向量; 对于 $k>d$ 的文件, 则截断至统一的长度.

图 1 所示为 MalConv 恶意代码检测模型的网络结构.

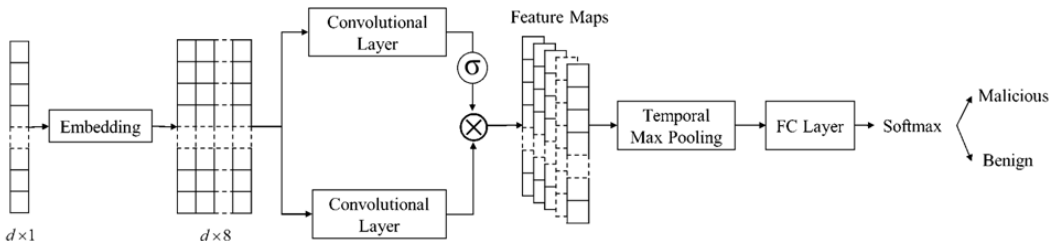


图 1 MalConv 恶意代码检测模型的结构

输入的字节序列首先经过一个嵌入层, 每个字节被映射为一个 8 维的向量, 作为卷积层的 8 个输入通道; 之后, 将 8 个输入通道分成两个 4 维通道, 分别输入两个设置了 *ReLU* 单元和 *Sigmod* 激活函数的卷积层, 并将两个卷积层的输出通过 Gating 方法[42]进行结合; 得到的值经过一次局部最大池化后输入一个带 *ReLU* 激活单元的全连接层; 最后, 通过一个 *Softmax* 层输出恶意或良性的分类结果.

2.2 对抗样本攻击模型

恶意代码对抗样本攻击方法根据攻击者是否掌握模型信息, 可以分为黑盒攻击和白盒攻击两类. 在白盒攻击场景下, 攻击者可以获取关于模型参数和结构等相关信息, 有更强的攻击能力, 也对防御方法有更高的要求. 因此, 本文主要研究白盒场景下的恶意代码对抗样本攻击的防御方法.

2.2.1 针对 PE 文件的修改

攻击者为了向恶意代码中添加扰动以影响检测模型的决策, 需要对输入检测模型的恶意代码 PE 文件的字节数据进行修改. 由于 PE 文件自身的特性, 文件的不同部分之间有较强的耦合性, 任意字节的修改都有可能造成程序运行错误. 在文件原本的功能不能被修改的前提下, 只针对某一部分的二进制数据进行修改往往比较困难. 为了保证恶意代码原有的恶意功能不被破坏, 已有的针对端到端恶意代码检测模型的对抗样本生成方法一般不会改动数据段、代码段等主要的功能部分, 而是通过修改 PE 文件的文件头、文件尾以及节与节之间的未用区域, 来向恶意代码样本中添加扰动. PE 文件的结构如图 2 所示, 图中标黄区域为对抗样本的扰动位置.

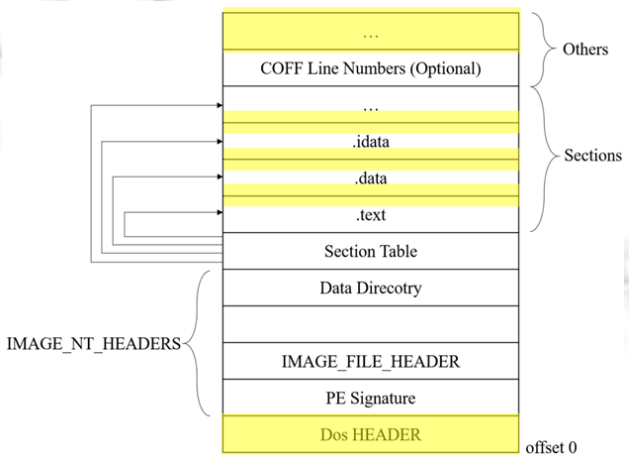


图 2 PE 文件结构及扰动位置

2.2.2 恶意代码对抗样本生成方法

Goodfellow 等人[14]在 2015 年提出了基于一步梯度的对抗样本生成算法 FGSM. 我们首先介绍 FGSM 算法, 在本节的后面部分将介绍针对恶意代码的对抗样本生成方法, 这些方法都是基于 FGSM 算法或改进的 FGSM 算法来生成扰动字节.

对抗样本生成算法对单一像素 z 的修改可以描述为

$$\tilde{z} = z + \delta \tag{1}$$

其中, δ是添加的扰动变量, $\tilde{z}$ 是扰动后像素的值. 已知攻击模型 g_θ, 其中, θ是固定的参数, FGSM 对抗样本生成算法可以表示为

$$\tilde{z} = z + \varepsilon \cdot sign(\nabla_z \ell(\tilde{z}, y : \theta)) \tag{2}$$

其中, y 是目标类别, ℓ是损失函数, ε是允许的最大扰动量.

FGSM 算法不能直接用于恶意代码对抗样本的生成, 因为构成恶意代码样本的字节值是离散的量, 无法进行梯度计算. 现有方法的思路是, 在嵌入空间中进行扰动. 第 2.1 节中介绍的恶意代码检测模型通过一个嵌

入层将每个字节映射到一个 8 维向量, 组成一个嵌入空间 $M \in R^{256\times 8}$. 生成对抗样本时, 对字节对应的 M 中的向量通过公式(2)进行修改, 并将修改后的向量 $\tilde{z}$ 映射到字节值. 修改后的向量 $\tilde{z}$ 如果不在嵌入空间 M 中, 则计算 M 中所有向量与 $\tilde{z}$ 的距离, 选择与 $\tilde{z}$ 距离最小的向量, 并将其对应的字节值作为修改后的字节值.

我们总结了现有的针对端到端恶意代码检测模型的对抗样本攻击方法, 根据生成对抗样本时在文件中添加扰动字节的位置, 将对抗样本生成方法分为如下 3 类.

(1) 在文件尾添加扰动. 对于文件长度不超过模型输入限制的恶意样本, 在恶意代码样本的文件末尾追加扰动字节是最直接的扰动方法. 文献[18]提出的方法就是通过在文件末尾追加扰动字节来生成恶意代码对抗样本, 其生成扰动字节的思想与 FGSM 算法基本一致. 该方法首先在文件末尾追加一定数量的任意字节, 并保证修改后的文件长度不超过模型的输入限制; 之后, 计算目标结果对嵌入空间向量的梯度, 并将追加的每一个字节替换为最接近梯度下降方向的向量对应的字节. 通过多次对以上过程的迭代生成对抗样本. 此类方法由于对添加扰动字节的数量限制较小, 因此可以添加较多的扰动字节, 扰动程度最大;

(2) 在文件中间位置插入扰动. 文献[19]提出一种基于上文介绍的 FGSM 算法的恶意代码对抗样本生成方法, 并且采用了在 PE 文件中的未用区域插入扰动字节的方法. 未用区域是指 PE 文件中一些未被程序使用的字节, 一般是在编译器编译过程中产生的, 这些未用字节一般存在于节与节之间以及节表与第一个节之间等位置. 如图 3 所示是 64 进制格式的 PE 文件中的未用字节, 修改这些未用字节不会对文件的功能产生影响;

(3) 修改文件头添加扰动. 文献[34]提出一种通过修改 PE 文件头来向恶意代码样本中添加扰动的方法, 并分析了基于深度学习的恶意代码检测模型, 认为虽然 PE 文件头部可以修改的字节数较少, 但对文件头的修改可以显著影响模型的决策. 该方法借鉴了文献[18]中生成扰动字节的方法, 通过向 DOS 头添加扰动生成对抗样本. 研究发现, 在 PE 文件的 DOS 头中, 除了固定字段“MZ”以及指向 PE 头的偏移量外, 其余字节的修改不会影响文件的正确运行, 所以可以通过修改这些字节添加扰动. 由于 DOS 头中可以修改的字节较少, 因此这类方法对恶意代码的扰动程度在 3 类方法中最小.

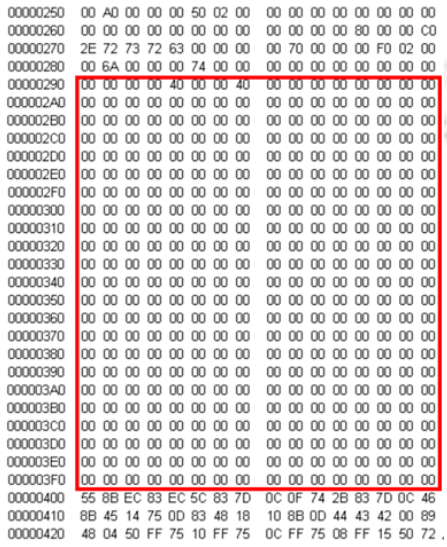


图 3 PE 文件中的未用字节

3 基于模型解释的对抗样本检测算法描述

通过对现有的恶意代码对抗样本攻击方法的总结与分析, 我们发现对抗样本的扰动会增加恶意代码样本

的文件头、文件尾及节间区域的字节对模型输出良性分类结果置信度的梯度值, 而这些部分在正常样本中信息量较低, 会导致正常良性样本和对抗样本中, 对模型决策起重要作用的字节在文件中的位置存在差异. 基于这一发现, 本文提出一种恶意代码对抗样本检测方法, 利用模型解释技术, 分析被恶意代码检测模型判定为良性的样本中每个字节对模型决策的贡献度, 构建样本的贡献度分布特征, 以此为依据识别对抗样本.

3.1 总体描述

基于模型解释的恶意代码对抗样本检测算法的流程分为构建和检测两个阶段, 如图 4 所示.

- 在构建阶段: 第 1 步, 通过对恶意代码检测模型的可解释性分析, 提取被模型判定为良性的输入样本中各部分对模型决策的贡献度, 构成贡献度分布向量; 第 2 步, 提取良性样本集中所有良性样本的贡献度分布向量, 组成良性样本贡献度分布向量集; 第 3 步, 基于良性样本的贡献度分布向量集构建一个异常检测器;
- 在检测阶段, 对于输入恶意代码检测模型的样本, 如果模型的检测结果为良性, 则将该样本输入构建好的异常检测器中, 分析其贡献度分布是否存在异常, 从而判断其是否为对抗样本.

该方法可以作为一个单独的模块附加在恶意代码检测模型后, 不需要对恶意代码检测模型的结构进行修改, 也不需要重新训练模型.

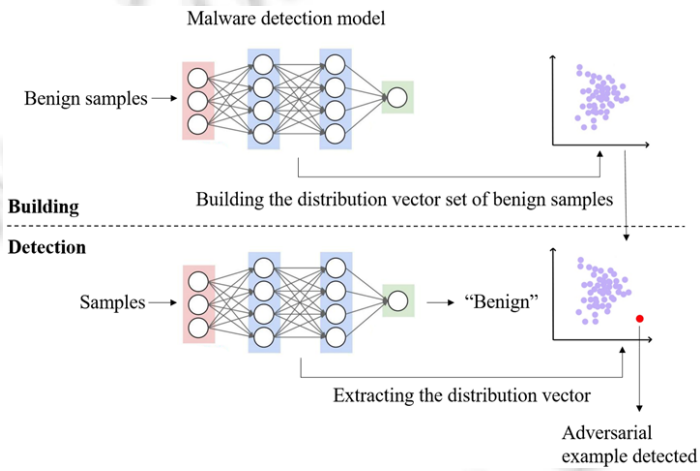


图 4 恶意代码对抗样本检测流程

3.2 恶意代码检测模型输入样本贡献度分布提取

对训练好的恶意代码检测模型, 我们引入模型解释方法来分析模型的决策过程. 对于输入模型的 PE 文件中的每个字节, 计算其对模型做出良性分类决策的贡献程度. 之后, 对字节按 PE 文件结构进行划分, 得到贡献度在 PE 文件中的分布情况.

3.2.1 基于模型解释的字节贡献度提取

本文以第 2 节中提到的恶意代码检测模型为研究对象, 引入基于 Grad-CAM[43]的卷积神经网络模型解释方法分析模型的决策依据, 提取模型输入样本中各字节的贡献度.

基于模型解释的字节贡献度提取方法如图 5 所示, 按照如下步骤进行.

(1) 以卷积层的输出为目标层, 提取 Gating 输出的 k 通道特征图 A^k 为目标层输出;

(2) 给定输出类别 c(例如良性), 求 *Softmax* 对指定类别的输出 y^c 对目标层输出的梯度;

(3) 对计算出的梯度, 通过求其全局平均池化得到每一个特征图的权重:

$$w_k = \frac{1}{n}\sum_i \frac{\partial y^c}{\partial A_i^k} \tag{3}$$

其中, i 表示特征图中的每一维;

(4) 最终, 根据权重对特征图求加权平均, 并通过一个 $ReLU$ 层过滤掉负梯度带来的噪声, 得到的贡献度向量为:

$$L^c = ReLU(\sum_k w_k^c A^k) \tag{4}$$

(5) 得到的贡献度向量 L^c 的尺寸与特征图 A 一致. 为了细化到字节粒度, 根据文件尺寸对 L^c 进行缩放, 得到每个字节的对应贡献度.

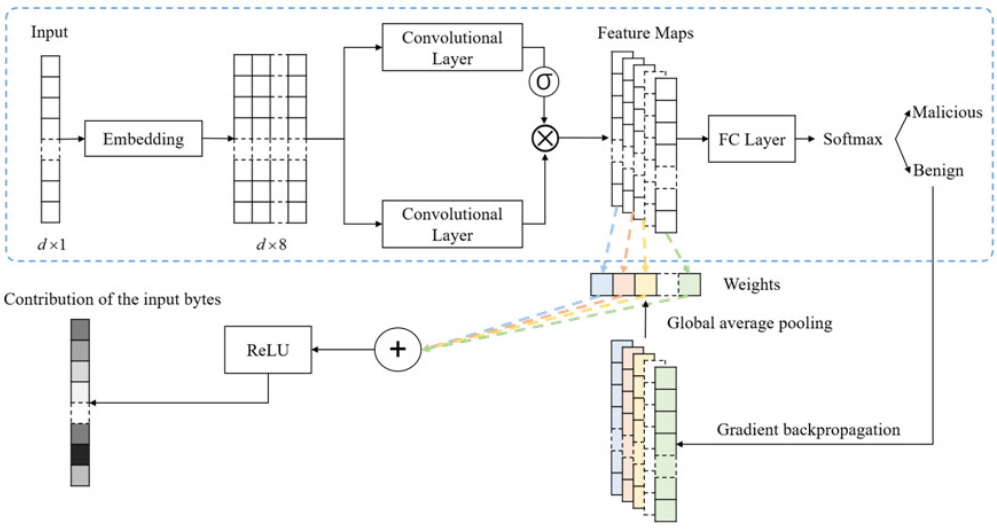
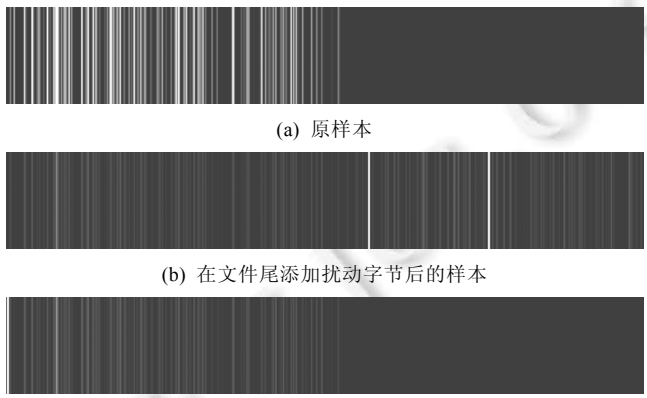


图 5 基于模型解释的字节贡献度提取方法

为了直观地表现字节贡献度提取方法获得向量的内容, 图 6 的例子展示了扰动前后样本的字节贡献度.



(a) 原样本

(b) 在文件尾添加扰动字节后的样本

(c) 在节间区域添加节后的样本

图 6 对抗扰动对字节贡献度的影响

图 6 为将字节贡献度向量转化为灰度图并放大宽度得到的结果, 我们选取了一个文件尺寸较大的样本, 从而能够较明显地表现字节贡献度在空间上的分布. 其中, 图 6(a)为正常良性样本的字节贡献度分布, 图 6(b)为在文件尾添加扰动后的字节贡献度分布, 图 6(c)为在节表与节间区域添加扰动后的字节贡献度分布. 可以看出: 三者之间有较为明显的差异, 扰动后的文件的字节贡献度在扰动区域较为集中. 此外, 为了避免选取样本存在特殊性导致结论不准确, 在第 3.2.2 节中, 我们给出了经过划分后得到贡献度分布向量的平均情况, 能更准确地反映扰动前后贡献度分布的差异.

3.2.2 贡献度分布向量的构建

为了分析贡献度在 PE 文件中不同部分的分布情况, 基于对 PE 文件的解析将字节贡献度进行分块, 构建样本的贡献度分布向量.

对于每一个输入模型的样本, 我们将其划分为若干子文件块, 构建一个贡献度分布向量 C, 向量的每一维 c_i 对应了 PE 文件中的一个子文件块, 其数值为这个部分全部字节的贡献度总和的占比, 向量每一维对应的子文件块划分见表 1.

表 1 贡献度分布向量

Index	Partition
1	Dos header
2	DOS stub
3	PE Signature
4	Image File Header
5	Image Optional Header
6	Section Header
7	Space between sections
8	Section 1
…	…
17	Section 10
18	Tail

对于一个模型输入样本, 首先计算样本文件所有字节的贡献度的总和 s; 之后, 对于贡献度分布向量中的每一项 c_i, 计算对应子文件块中所有字节的贡献度的和 s_i, 通过计算 s_i 与 s 的比值得到 c_i. 节区域是 PE 文件的主要数据区域, 占用了文件的大部分空间, 为了体现贡献度在节中的空间分布, 将节进一步分块. 例如表 1 中, 将节按照字节数均分为 10 个子块. 对于有不止一个节的文件, 对多个节的贡献度分布向量求平均值, 合并为一个节.

在第 2.2 节中已经分析过, 现有的恶意代码对抗样本攻击方法主要分为 3 类, 我们复现了文献[18]、文献[19]以及文献[34]中的对抗样本攻击方法, 并选取了 500 个恶意代码分别使用上述 3 种对抗样本攻击方法进行扰动, 得到 3 个对抗样本集. 首先对每个对抗样本提取贡献度分布向量, 之后对每个对抗样本集分别求贡献度向量的平均值, 得到每个对抗样本集的各子文件块平均贡献度值如图 7 所示.

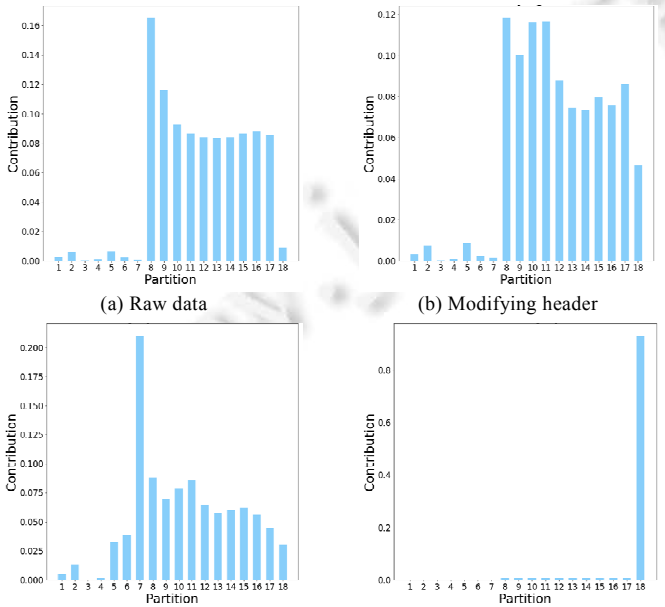


图 7 扰动前后的平均贡献度分布

每个子图的横坐标对应了表 1 中 PE 文件子文件块的索引号, 其中, 图 7(a)子图为原样本集的平均贡献度分布, 图 7(b)–图 7(d)子图分别为用 3 种对抗样本攻击方法对原样本集扰动后的对抗样本集的贡献度分布. 从图中可以看出, 对抗样本中扰动位置的贡献度值会显著高于其他文件位置的贡献度值. 例如: 在文件尾添加扰动字节的攻击方法得到的对抗样本中, 文件尾部的贡献度值要显著高于其他文件位置; 在节间插入扰动字节得到的对抗样本中, 节间区域的贡献度值要显著高于其他文件位置. 其中, 在文件头添加扰动的攻击方法由于修改的字节数较少, 对样本贡献度分布造成的影响不明显.

3.3 贡献度分布异常检测

在真实场景中,对抗样本攻击方法一般是未知的, 无法得知攻击者在生成恶意代码对抗样本时在文件中添加扰动字节的位置, 所以本文基于无监督的异常检测方法研究对抗样本的检测.

我们采用了基于孤立森林[44]的异常检测方法, 以贡献度分布的异常为根据识别对抗样本. 首先, 基于良性样本集构造贡献度分布向量集 X, X 中包含了从良性样本集所有良性样本中提取的贡献度分布向量, 利用 X 构造孤立森林来识别异常样本.

构造异常检测模型时, 第 1 步是训练孤立树. 对于树中的节点 T, 如果 T 是内部节点, 则 T 有两个子节点 (T_l,T_r). 在训练时, 从数据向量中随机选择一个维度 q 并设置一个分割点 p 进行分割操作. 对于当前样本, 若 $q<p$, 则样本属于 T_l; 反之, 则属于 T_r. 对于一个样本 x, x 在孤立树中的路径长度 $h(x)$为样本 x 按照节点分割从根节点到叶子节点所经过的边的数量. 下面是算法的具体描述.

(1) 已有 $X=\{x_1,\ldots,x_n\}$, 并且$\forall x_i\in X$, $x_i=(x_{i1},\ldots,x_{id})$, 从 X 中随机抽取φ个样本构成 X 的子集 X'放入根节点;

(2) 从 d 个维度中随机选定一个维度 q, 并在维度 q 中随机选择一个分割点 p, 选择的 p 应该满足:

$$\min(x_{iq},x_{iq}\in X')<p<\max(x_{iq},x_{iq}\in X') \tag{5}$$

(3) 根据切割点 p 产生的超平面, 将当前的样本集中的样本分别放入两个子节点中;

(4) 递归执行步骤(1)和步骤(2), 直至所有叶子节点中都只有一个样本, 或孤立树的高度已达到阈值;

(5) 循环步骤(1)到步骤(4), 直至生成 t 个孤立树;

(6) 对于每一个样本 x_i, 计算 x_i 在森林中所有孤立树中的平均路径长度并进行标准化处理. 最终得到的异常值分数为

$$s(x,\varphi)=2^{\frac{E(h(x))}{c(\varphi)}} \tag{6}$$

其中, $c(\varphi)$为给定样本数φ时的平均样本长度, 用于做标准化处理. $c(\varphi)$的计算公式为

$$c(\varphi)=\begin{cases}2H(\varphi-1)-2(\varphi-1)/\varphi, & \varphi>2\\ 1, & \varphi=2\\ 0, & \text{otherwise}\end{cases} \tag{7}$$

其中, $H(x)$是调和级数.

完成上述步骤后, 得到样本 x_i 的异常值分数. 如果异常值分数接近 1, 说明样本 x_i 的路径长度要远小于数据集的平均路径长度, 则样本 x_i 为异常点, 将其判断为对抗样本.

4 实验结果与分析

本节第 1 部分介绍了实验的设置, 在第 2 部分介绍对抗样本检测方法对不同对抗样本攻击的检测效果, 第 3 部分对本文方法与对抗训练的防御效果进行了对比. 本节的最后部分对实验结果进行了讨论.

4.1 实验设置

实验平台: Xeon Silver 4214 2.20GHzx12 (CPU), RTX 2080Ti 11GBx2 (GPU), 125 GB 内存, Ubuntu 18.04 (操作系统), Python 3.7,Pytorch 1.4.0 (深度学习框架)

实验所使用的数据包含了从 vxheaven 网站公开的恶意代码数据集中获得的 PE 文件格式恶意代码样本以

及从 Windows 系统文件中获得的良性样本, 从中随机选取并划分为表 2 所示的 4 个数据集. 4 个数据集的样本相互独立, 不存在交集. 其中: 数据集 1 作为训练集, 用于训练恶意代码检测模型; 数据集 2 作为测试集, 用于测试恶意代码检测模型的分类效果; 数据集 3 用于测试对抗样本对恶意代码检测模型的攻击能力以及对抗样本检测方法对对抗样本的检测效果; 数据集 4 用于训练对抗样本检测器, 由于对抗样本检测器可以单独训练, 不依赖于恶意代码检测模型, 因此这里使用的数据集 4 与数据集 1 也是隔离的.

表 2 实验中使用的数据集

数据集	良性样本数	恶意样本数
1	5 000	5 000
2	1 000	1 000
3	500	500
4	4 000	0

实验使用的恶意代码检测模型复现了文献[10]中的 Malconv 方法, 网络结构设置见表 3, 其中, 第 3 层为两个并行的卷积层. 训练中采用了批训练的方式进行了 10 轮训练, 训练结束后, 恶意代码检测模型在数据集 2 中的准确率为 96%, 与文献[10]中的实验结果相近.

表 3 恶意代码检测模型结构

序号	类型	滤波器尺寸/步长	输出尺寸
1	输入	–	2000000×1
2	Embedding	–	2000000×8
3	*Conv*1	500×1/500	4000×1×128
3	*Conv*2	500×1/500	4000×1×128
4	Gating	–	4000×1×128
5	*ReLU*1	–	4000×1×128
6	Pool	4000×1/4000	128×1
7	Fully connected	–	128×1
8	*ReLU*2	–	128×1
9	Fully connected	–	2×1
20	*Softmax*	–	2×1

4.2 针对不同攻击方法的防御效果

对于数据集 3, 按照第 3.2.3 节中描述的方法, 对其中的恶意样本分别使用 3 种攻击方法进行扰动, 并分别得到 3 个包含对应对抗样本的测试集. 此外, 由于 3 种对抗样本攻击方法对于 PE 文件的扰动位置不相重叠, 我们综合了 3 种方法进行扰动, 得到 1 个额外的测试集用于对防御效果的测试, 因此, 最终共有 4 个包含对抗样本的测试集. 在第 2.2 节中已进行讨论, 使用的 3 种攻击方法不会影响 PE 文件的运行逻辑. 为了进一步验证实验中使用的方法对 PE 文件的修改是问题空间中的有效修改, 我们使用 Cuckoo 沙盒测试工具对可执行文件修改前后的运行状况和行为进行了对比. 由于一些可执行文件需要在特定的条件下才可执行, 因此选取了 50 个在修改前可以运行的样本进行了测试. 测试中, 主要对修改前后可执行文件的 API 调用、文件操作、网络连接、注册表行为进行了对比. 测试样本的对比结果显示, 实验中使用的对抗样本生成方法不会对可执行文件的运行状况和功能造成影响.

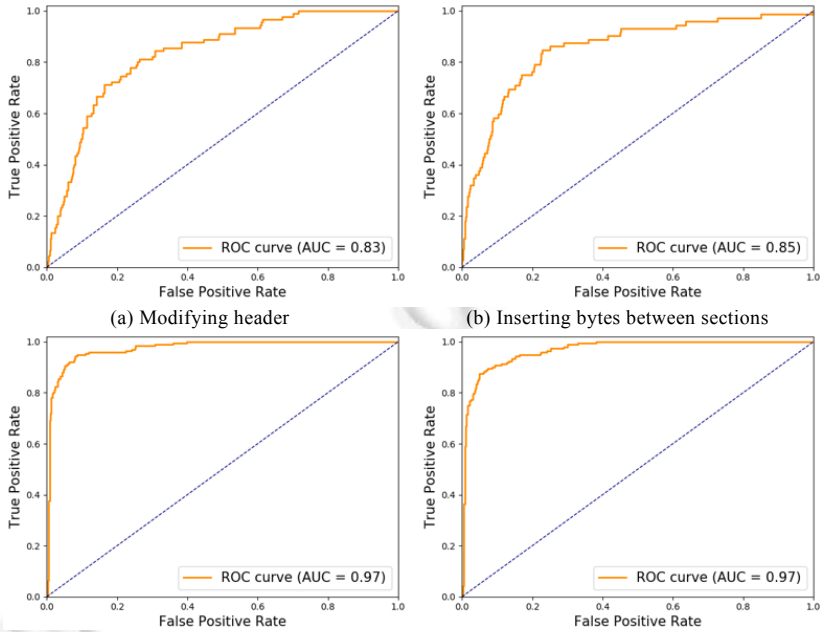
在测试方法的防御效果前, 我们首先分析了每种攻击方法对恶意代码检测模型的攻击能力, 通过模型的误分类率来评估. 误分类率定义为: 经过对抗扰动后, 被错误分类的样本数占扰动前总的恶意样本数的比例. MalConv 模型对上述 4 个对抗样本集进行测试的结果见表 4. 实验结果中, 4 种攻击方法的攻击能力从高到低依次是: 综合扰动、文件尾扰动、节间扰动和文件头扰动.

表 4 不同攻击方法下恶意代码检测模型的误分类率

	文件头扰动	节间扰动	文件尾扰动	综合扰动
错误分类率(%)	43.6	51.2	66.8	71.4

提取数据集 4 中所有样本的贡献度分布向量, 并按照第 3.3 节中的方法训练对抗样本检测器. 得到的对抗

样本检测器在的 4 个测试集上的检测结果如图 8 所示. 图中的图 8(a)~图 8(d)子图分别对应了检测器对包含文件头扰动、节间扰动、文件尾扰动、综合扰动对抗样本的测试集的检测结果. 分析可得: 检测器对文件尾扰动和综合扰动的检测效果最好, 对节间扰动对抗样本的检测效果略好于文件头扰动.



(a) Modifying header (b) Inserting bytes between sections

(c) Padding bytes (d) Integrated perturbation

图 8 检测方法对不同对抗样本攻击方法的检测结果

4.3 防御效果的对比

我们以第 4.2 节中生成的 4 个对抗样本集为输入, 对防御方法的防御效果进行测试. 除了本文提出的对抗样本检测方法, 我们选择了对抗训练作为对比的防御方法. 此外, 通过本文方法得到的对抗样本检测器可以作为独立的模块附加在恶意代码检测模型后. 在实验中, 我们将对抗训练后得到的模型附加了对抗样本检测模块, 并将这种组合作为一种新的防御方法, 与单独使用两种方法的防御能力进行了对比. 实验中测试的 3 种方法分别为: 原模型附加对抗样本检测模块、对抗训练的模型、对抗训练的模型附加检测模块.

在进行对抗训练时, 我们参考了文献[22,23]的实验设置, 训练过程分为 3 步: 首先, 按照第 4.1 节中的设置训练恶意代码检测模型; 之后, 以训练好的模型为目标, 生成与测试用对抗样本集等量的 4 类对抗样本; 最后, 将生成的对抗样本加入训练集重新训练模型. 其中, 生成对抗样本时所使用的恶意代码样本为随机选出的与其他样本集不重复的样本.

4.3.1 防御方法的抵抗率

实验中, 采用“抵抗率”作为评估对抗样本防御方法防御能力的指标, 抵抗率定义为无法成功攻击模型的对抗样本数占总的对抗样本数的比例. 最终得到的实验结果见表 5.

表 5 防御方法对不同对抗样本攻击方法的抵抗率

模型	抵抗率(%)			
	文件头扰动	节间扰动	文件尾扰动	综合扰动
本文方法	90.38	94.84	**98.12**	**97.42**
对抗训练	90.14	82.63	51.41	53.76
对抗训练结合本文方法	**92.49**	**95.54**	97.89	96.95

从表 4 中可以看出, 3 种防御方法对实验生成的对抗样本都有一定的防御效果. 本文方法对 4 种对抗样本

攻击的抵抗率要整体高于对抗训练方法, 其中: 对文件尾扰动的攻击方法抵抗率最高, 达到了 98.12%; 对文件头扰动的攻击方法抵抗率为 90.38%, 相对较低. 此外, 结合了两种防御方法得到的新模型对文件头扰动和节间扰动这两类攻击方法的抵抗率要高于单独使用两种防御方法, 对文件尾扰动和综合扰动这两类攻击方法的抵抗率略低于单独使用本文方法, 但高于对抗训练方法, 这说明将本文方法与其他防御方法结合后可以在一定程度上提升防御能力.

对实验结果的分析可以发现: 对抗训练的防御效果受扰动位置的影响较大, 其中, 对文件尾扰动的对抗样本防御效果相对较差. 针对这一问题, 我们分析了对抗训练中使用的对抗样本数量对防御效果的影响, 随机去除部分训练中使用的文件尾部扰动的对抗样本, 并分别以文件尾扰动总样本数的 10%到 100%的样本进行对抗训练, 得到模型对文件尾扰动对抗样本的抵抗率如图 9 所示. 从图中可以看出: 在曲线后半段, 对抗训练使用的对抗样本数量的增多对抵抗率的提升不显著, 在样本数 50%至 70%区间抵抗率略有降低.

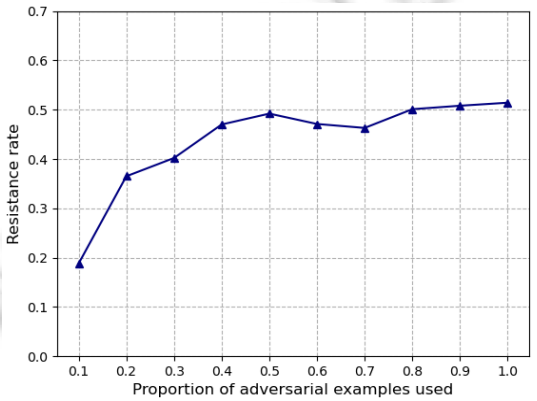


图 9 对抗训练对文件尾扰动的抵抗率

进一步分析发现: 实验中的几种攻击方法除了扰动位置的不同, 样本中加入的扰动字节的数量也不同. 因此, 我们推测: 对抗训练防御效果的差异是插入扰动字节数量的不同造成的. 扰动字节的数量取决于文件的不同位置允许修改的空间的大小: 文件头部自身的空间比较小, PE 文件的 DOS 头部的总容量为 64 字节, 根据文献[34]的研究, 其中可修改的数据有 60 个字节; 对于 PE 文件的节间区域, 取决于编译中的对齐情况, 存在的空白空间是变化的, 但每一块空白空间不会超过编译器默认的对齐值, 实验中测得的单个文件的平均可扰动空间为 846 个字节; 文件尾可以插入扰动字节空间较大, 主要受模型输入的样本尺寸及扰动效率的限制, 参考文献[18]的设置, 实验中对每个样本添加了 10 000 个扰动字节.

由此可见, 文件尾扰动及综合扰动添加的扰动字节数要远多于文件头和节间扰动. 这会产生较大的样本空间, 使用相同的扰动字节序列生成方法进行对抗训练, 难以获得与另外两种扰动攻击相同的抵抗率. 因此, 对于文件尾扰动的对抗训练防御, 可能需要更有效的对抗样本生成方法的参与, 才能使防御效果有较大提升. 而对于本文方法, 文件尾扰动和综合扰动添加的扰动字节较多, 会使得对抗样本的贡献度分布在异常检测中表现出更明显的差异. 因此, 对文件尾扰动对抗样本的抵抗率要高于另外两种扰动获得的对抗样本.

添加扰动字节的数量是恶意代码对抗样本扰动强度的主要体现, 因为不同于图像对抗样本, 生成恶意代码对抗样本的过程不会对字节值的修改幅度进行限制. 文件头可以添加的扰动字节的数量较少, 而节间区域可插入的空间是不确定的, 因此, 为了研究添加扰动字节的数量对恶意代码对抗样本的攻击和防御的影响, 我们单独分析了文件尾扰动的情况, 对添加的扰动字节数量变化时攻击和防御效果的变化情况进行了测试, 得到的结果如图 10 所示. 可以看出: 随着添加的扰动字节的数量的增加, 对抗样本的攻击能力有明显的提升, 但同时, 对抗样本检测器的检测率也有升高.

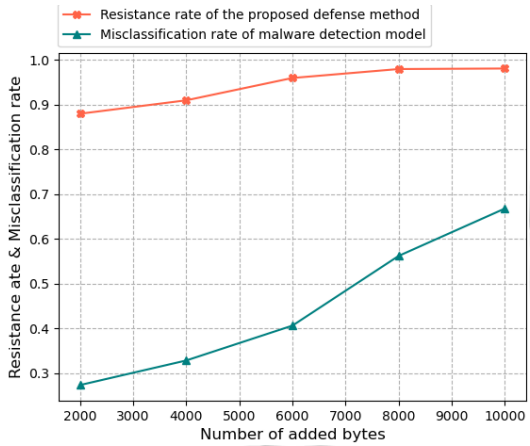
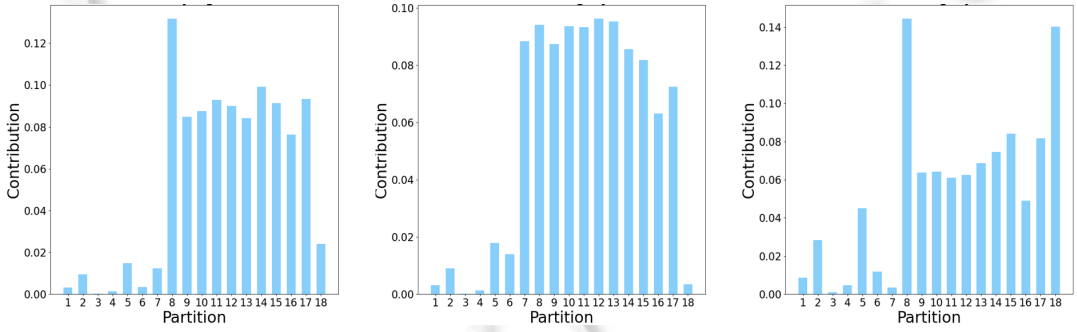


图 10　扰动字节数量变化时，模型的误分类率及防御方法的抵抗率

此外，通过对实验结果的分析可以发现：对于本文提出的对抗样本检测方法，仍有少部分对抗样本可以逃逸检测．我们对这部分对抗样本提取了贡献度分布向量，并得到其平均贡献度分布如图 11 所示．通过与图 7 的对比可以发现：相较于全体对抗样本，成功逃逸检测的样本中，贡献度分布表现出来的对抗样本特征有所减弱，整体的分布更接近于正常样本，具体表现为：扰动区域如文件尾和节间的贡献度所占的比例有明显的下降，节内区域的贡献度所占比例升高．这说明本文的方法并不能完全覆盖恶意代码检测模型决策存在的漏洞，仍然存在一些对抗样本在生成过程中搜索到了这些未覆盖的区域．在这些区域的对抗样本能够绕过恶意代码检测模型，同时，在功能无关区域的贡献度不明显．但实验中仅少部分样本能够逃逸检测，说明本文提出的方法是有效的；且由于本方法作为独立的模块，可以与对抗训练等其他对抗样本防御方法相结合进行补充，进一步增加对抗样本生成的难度．



(a) Modifying header　(b) Inserting bytes　(c) Padding bytes

图 11　未被识别的对抗样本的平均贡献度分布

4.3.2　防御方法对模型分类正常样本的影响

对抗样本防御提升了恶意代码检测模型抵抗对抗样本攻击的能力，同时，由于对抗样本防御方法参与了模型的分类过程，也会对模型分类其他正常样本产生影响．正常样本即非对抗样本，包括正常良性样本及正常恶意样本．为了研究对抗样本防御方法对模型分类效果的影响，选取了准确率(accuracy)、召回率(recall)、精确率(precision)、$F1$-score 作为评价指标，对采取了对抗样本防御方法前后的模型分类效果进行了评估，测试集中不包含对抗样本．作为对比，我们也测试了对抗训练和结合了两种方法的防御模型的分类效果．实验结果如表 6 所示．

由表 6 可以看出，3 种对抗样本防御方法都会造成模型分类准确率的降低．其中，本文提出的对抗样本检测方法与对抗训练方法对模型分类正常样本的影响相近：原模型附加对抗样本检测模块后，精确率下降到了

0.915, 召回率有少量提升; 原模型经过对抗训练后, 模型精确率下降到了 0.916, 召回率提升至 0.959. 结合了对抗训练和本文方法的综合防御方法对模型分类准确率影响最大, 在使用了综合防御方法后, 模型分类效果的 4 项评估指标都有所下降, 但仍在 0.9 以上.

表 6 不同防御方法对模型分类正常样本的影响

测试集	Accuracy	Precision	Recall	*F*1-score
原模型	0.960	0.975	0.946	0.960
本文方法	0.931	0.915	0.949	0.932
对抗训练	0.939	0.916	0.959	0.937
对抗训练结合本文方法	0.913	0.904	0.923	0.914

通过对实验结果的分析发现, 对抗样本检测模块对模型分类正常样本产生的影响来自于两个方面: 一是对良性样本的误报, 即被原模型识别为良性的良性样本在原模型附加检测模块后被误识别为对抗样本; 二是对假良性样本的重识别, 即被原模型识别为良性的恶意样本在原模型附加检测模块后被识别为对抗样本. 我们单独对这两类样本进行了测试, 原模型附加对抗样本检测模块后, 对假良性样本的重识别率为 8.89%, 对良性样本的误报率为 6.62%.

对良性样本的误报及假良性样本的重识别实际上反映了恶意代码检测模型的不足: 对于一些良性样本, 恶意代码检测模型仅依靠文件头、文件尾等区域提取的特征做出了决策, 并未从主要的数据区域提取到帮助决策的特征, 因此导致贡献度分布于对抗样本相似, 从而产生了对良性样本的误报; 同样地, 对于一些恶意代码样本, 恶意代码检测模型更多地关注了节以外区域的特征, 从而造成了对恶意代码的漏报, 同时也导致了样本的贡献度分布相似于对抗样本, 因此能够被对抗样本检测器重识别.

4.4 实验结果的讨论

对实验结果的分析表明: 本文提出的恶意代码对抗样本检测方法能够有效防御现有的各类恶意代码对抗样本攻击方法, 且防御效果显著优于对抗训练方法. 此外, 本文的方法也存在一些问题: 第一, 本文的方法存在大部分对抗样本防御方法的共性问题, 即对恶意代码检测器分类正常样本会造成一定的影响, 实验测试得到的实际影响程度与对抗训练方法相近, 处在可接受的范围内; 第二, 现有的 PE 文件恶意代码对抗样本攻击方法通过插入扰动字节生成对抗样本, 受 PE 文件格式的限制, 只能在有限的区域进行扰动, 因此可以通过分析贡献度分布对其进行有效的检测, 但是不排除未来会出现不受扰动位置限制的扰动方法, 如通过修改代码语义等更精细的操作进行扰动. 对于这些扰动方法, 仅基于贡献度分布特征识别对抗样本的方法不再有效, 此时就需要结合其他防御方法加强防御能力. 本文的方法可以作为独立的模块附加在恶意代码检测模型后, 因此可以很方便地与其他对抗样本防御方法进行结合, 实验结果也验证了组合方法的有效性.

实验中的良性样本误报问题及假良性样本重识别问题, 从侧面反映了基于深度学习的端到端恶意代码检测技术存在一些不足. 通过对贡献度分布的分析我们发现: 对于一些正常 PE 文件样本, 恶意代码检测模型并没有以样本中主要的数据区域表现出的特征为主要的决策依据, 而是以文件头、文件尾这些信息含量较低的区域表现出的特征为主要决策依据, 恶意代码检测模型在分类这些样本时并未提取到有效特征, 从而导致了错误的分类结果, 这为改进恶意代码检测模型提供了思路.

5 总 结

对抗样本技术揭露了深度学习的脆弱性, 一些基于深度学习模型的智能系统如人脸识别系统、自动驾驶系统、恶意代码检测系统等因此面临安全风险. 本文研究基于深度学习的恶意代码检测系统的对抗样本防御问题. 不同于图像对抗样本, 生成恶意代码对抗样本时, 添加的扰动不能影响恶意代码的原有功能, 因此现有的 PE 文件恶意代码对抗样本攻击方法一般通过在文件尾、节与节之间等不影响文件功能的区域添加扰动字节来生成对抗样本. 本文基于模型解释技术, 提出了一种恶意代码检测模型决策依据的分析方法, 基于对抗样本的贡献度分布特征对恶意代码对抗样本进行准确识别. 实验结果表明, 本文提出的对抗样本检测方法能

够以较高的抵抗率防御恶意代码对抗样本攻击. 相较于对抗训练等对抗样本防御方法, 本文的方法采用无监督的方式构建检测器, 不需要提前获取对抗样本数据来训练模型, 能够防御多种对抗样本攻击方法; 此外, 本文提出的对抗样本检测方法不需要重新训练模型或者修改模型结构, 能够更高效地应用于已部署的恶意代码检测模型, 更符合实际应用的需求. 未来的工作之一是: 研究如何利用模型解释方法评估基于深度学习的恶意代码检测模型特征提取的有效性, 以建立更健壮的模型. 此外, 我们将对本文提出的对抗样本检测方法进行改进和扩展, 研究如何将本文方法推广至 ELF 等相似文件格式恶意代码对抗样本的检测. 我们还将继续研究模型解释方法与非端到端模型的结合, 以将本文方法适用于更多的模型结构, 并通过对模型解释和异常检测方法的优化提升对抗样本的检测效果.

References:

[1] He K, Zhang X, Ren S, *et al*. Deep residual learning for image recognition. In: Proc. of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition. IEEE, 2016. 770−778. [doi: 10.1109/CVPR.2016.90]

[2] Amodei D, Ananthanarayanan S, Anubhai R, *et al*. Deep speech 2: End-to-end speech recognition in english and mandarin. In: Proc. of the Int'l Conf. on Machine Learning. ACM, 2016. 173−182.

[3] Young T, Hazarika D, Poria S, *et al*. Recent trends in deep learning based natural language processing. IEEE Computational Intelligence Magazine, 2018, 13(3): 55−75. [doi: 10.1109/MCI.2018.2840738]

[4] Shin ECR, Song D, Moazzezi R, *et al*. Recognizing functions in binaries with neural networks. In: Proc. of the 24th USENIX Security Symp. (USENIX Security 2015). USENIX Association, 2015. 611−626.

[5] Chua ZL, Shen S, Saxena P, *et al*. Neural nets can learn function type signatures from binaries. In: Proc. of the 26th USENIX Security Symp. (USENIX Security 2017). USENIX Association, 2017. 99−116.

[6] Zhang YQ, Dong Y, Liu CY, *et al*. Situation trends and prospects of deep learning applied to cyberspace security. Journal of Computer Research and Developmentrtret, 2018, 55(6): 1117−1142 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2018.20170649]

[7] Spafford EC. Is anti-virus really dead? Computers & Security, 2014, 44: iv. [doi: 10.1016/S0167-4048(14)00082-0]

[8] Abou-Assaleh T, Cercone N, Keselj V, *et al*. *N*-Gram-Based detection of new malicious code. In: Proc. of the 28th Annual Int'l Computer Software and Applications Conf. IEEE, 2004. 41−42. [doi: 10.1109/CMPSAC.2004.1342667]

[9] Rieck K, Holz T, Willems C, *et al*. Learning and classification of malware behavior. In: Proc. of the Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2008. 108−125. [doi: 10.17877/DE290R-2041]

[10] Raff E, Barker J, Sylvester J, *et al*. Malware detection by eating a whole EXE. In: Proc. of the Workshops at the 32nd AAAI Conf. on Artificial Intelligence. AAAI, 2018. 268−276.

[11] Ren Z, Wu H, Ning Q, *et al*. End-to-End malware detection for Android IoT devices using deep learning. Ad Hoc Networks, 2020, 101(1): 1−11. [doi: 10.1016/j.adhoc.2020.102098]

[12] Amin M, Tanveer TA, Tehseen M, *et al*. Static malware detection and attribution in android byte-code through an end-to-end deep system. Future Generation Computer Systems, 2020, 102(1): 112−126. [doi: 10.1016/j.future.2019. 07.070]

[13] Szegedy C, Zaremba W, Sutskever I, *et al*. Intriguing properties of neural networks. arXiv:13126199, 2013.

[14] Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. arXiv:11041070, 2014.

[15] Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: Proc. of the 2017 IEEE Symp. on Security and Privacy (S&P). IEEE, 2017. 39−57. [doi: 10.1109/SP.2017.49]

[16] Papernot N, McDaniel P, Jha S, *et al*. The limitations of deep learning in adversarial settings. In: Proc. of the 2016 IEEE European Symp. on Security and Privacy (EuroS&P). IEEE, 2016. 372−387. [doi: 10.1109/EuroSP. 2016.36]

[17] Pierazzi F, Pendlebury F, Cortellazzi J, *et al*. Intriguing properties of adversarial ml attacks in the problem space. In: Proc. of the 2020 IEEE Symp. on Security and Privacy (S&P). IEEE, 2020. 1332−1349.

[18] Kolosnjaji B, Demontis A, Biggio B, *et al*. Adversarial malware binaries: Evading deep learning for malware detection in executables. In: Proc. of the 26th European Signal Processing Conf. (EUSIPCO). IEEE, 2018. 533−537. [doi: 10.23919/EUSIPCO. 2018.8553214]

[19] Kreuk F, Barak A, Aviv-Reuven S, *et al*. Deceiving end-to-end deep learning malware detectors using adversarial examples. arXiv:180204528, 2018.

[20] Yang JY. IDAE: Iterative denoising autoencoder based deep learning model enhancement mechanism against adversarial examples. Journal of Cyber Security, 2019, 4(6): 34−44 (in Chinese with English abstract).

[21] Gu S, Rigazio L. Towards deep neural network architectures robust to adversarial examples. In: Proc. of the NIPS Workshop on Deep Learning and Representation Learning. 2014.

[22] Grosse K, Papernot N, Manoharan P, *et al*. Adversarial examples for malware detection. In: Proc. of the European Symp. on Research in Computer Security. Springer, 2017. 62−79.

[23] Yang W, Kong D, Xie T, *et al*. Malware detection in adversarial settings: Exploiting feature evolutions and confusions in android apps. In: Proc. of the 33rd Annual Computer Security Applications Conf. 2017. 288−302.

[24] Shaham U, Yamada Y, Negahban S. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. Neurocomputing, 2018, 307(1): 195−204. [doi: 10.1016/j.neucom.2018.04.027]

[25] Madry A, Makelov A, Schmidt L, *et al*. Towards deep learning models resistant to adversarial attacks. In: Proc. of the 6th Int'l Conf. on Learning Representations. 2018.

[26] Saxe J, Berlin K. Deep neural network based malware detection using two dimensional binary program features. In: Proc. of the 2015 10th Int'l Conf. on Malicious and Unwanted Software (MALWARE). IEEE, 2015. 11−20.

[27] Hardy W, Chen L, Hou S, *et al*. Dl4md: A deep learning framework for intelligent malware detection. In: Proc. of the Int'l Conf. on Data Science (ICDATA). WorldComp, 2016. 61.

[28] Kumar R, Xiaosong Z, Khan RU, *et al*. Malicious code detection based on image processing using deep learning. In: Proc. of the 2018 Int'l Conf. on Computing and Artificial Intelligence. ACM, 2018. 81−85.

[29] Choi S, Jang S, Kim Y, *et al*. Malware detection using malware image and deep learning. In: Proc. of the 2017 Int'l Conf. on Information and Communication Technology Convergence (ICTC). IEEE, 2017. 1193−1195.

[30] Athiwaratkun B, Stokes JW. Malware classification with LSTM and GRU language models and a character-level CNN. In: Proc. of the 2017 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017. 2482−2486.

[31] Huang W, Stokes JW. MtNet: A multi-task neural network for dynamic malware classification. In: Proc. of the Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2016. 399−418.

[32] Pan WW, Wang XY, Song ML, *et al*. Survey on generating adversarial examples. Ruan Jian Xue Bao/Journal of Software, 2020, 31(1): 67−81 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5884.htm [doi: 10.13328/j.cnki.jos.005884]

[33] Chen X, Li C, Wang D, *et al*. Android HIV: A study of repackaging malware for evading machine- learning detection. IEEE Trans. on Information Forensics and Security, 2019, 15: 987−1001.

[34] Demetrio L, Biggio B, Lagorio G, *et al*. Explaining vulnerabilities of deep learning to adversarial malware binaries. arXiv:190103583, 2019.

[35] Hu W, Tan Y. Generating adversarial malware examples for black-box attacks based on GAN. arXiv:170205983, 2017.

[36] Rosenberg I, Shabtai A, Rokach L, *et al*. Generic black-box end-to-end attack against state of the art API call based malware classifiers. In: Proc. of the Int'l Symp. on Research in Attacks, Intrusions, and Defenses. Springer, 2018. 490−510.

[37] Papernot N, McDaniel P, Wu X, *et al*. Distillation as a defense to adversarial perturbations against deep neural networks. In: Proc. of the 2016 IEEE Symp. on Security and Privacy (S&P). IEEE, 2016. 582−597. [doi: 10.1109/SP.2016.41]

[38] Tramèr F, Kurakin A, Papernot N, *et al*. Ensemble adversarial training: Attacks and defenses. In: Proc. of the 6th Int'l Conf. on Learning Representations. ICLR, 2017. 1−20.

[39] Yang Y, Zhang G, Katabi D, *et al*. Me-Net: Towards effective adversarial robustness with matrix estimation. In: Proc. of the Int'l Conf. on Machine Learning. 2019. 7025−7034.

[40] Wang Q, Guo W, Zhang K, *et al*. Adversary resistant deep neural networks with an application to malware detection. In: Proc. of the 23rd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. 2017. 1145−1153. [doi: 10.1145/3097983.3098158]

[41] Liu N, Yang H, Hu X. Adversarial detection with model interpretation. In: Proc. of the 24th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. ACM, 2018. 1803−1811. [doi: 10.1145/3219819.3220027]

[42] Dauphin YN, Fan A, Auli M, *et al*. Language modeling with gated convolutional networks. In: Proc. of the Int'l Conf. on Machine Learning. ACM, 2017. 933−941.

[43] Selvaraju RR, Cogswell M, Das A, *et al*. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In: Proc. of the IEEE Int'l Conf. on Computer Vision. IEEE, 2017. 618−626. [doi: 10.1109/ICCV. 2017.74]

[44] Liu FT, Ting KM, Zhou ZH. Isolation forest. In: Proc. of the 8th IEEE Int'l Conf. on Data Mining. IEEE, 2008. 413−422. [doi: 10.1109/ICDM.2008.17]

附中文参考文献:

[6] 张玉清, 董颖, 柳彩云, 等. 深度学习应用于网络空间安全的现状、趋势与展望. 计算机研究与发展, 2018, 55(6): 1117−1142. [doi: 10.7544/issn1000-1239.2018.20170649]

[20] 杨浚宇. 基于迭代自编码器的深度学习对抗样本防御方案. 信息安全学报, 2019, 4(6): 34−44.

[32] 潘文雯, 王新宇, 宋明黎, 等. 对抗样本生成技术综述. 软件学报, 2020, 31(1): 67−81. http://www.jos.org.cn/1000-9825/5884.htm [doi: 10.13328/j.cnki.jos.005884]

田志成(1996−), 男, 硕士, 主要研究领域为网络空间安全.

乔延臣(1988−), 男, 博士, 助理研究员, CCF 专业会员, 主要研究领域为互联网体系结构, 网络空间安全.

张伟哲(1976−), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为网络空间安全, 云计算, 高性能计算.

刘洋(1988−), 男, 博士, 助理教授, CCF 专业会员, 主要研究领域为数据安全与隐私计算.