计算机科学
COMPUTER SCIENCE

基于遗传算法的恶意软件对抗样本生成方法

李坤, 郭威, 张帆, 杜加玉, 杨梅樾

引用本文

李坤, 郭威, 张帆, 杜加玉, 杨梅樾基于遗传算法的恶意软件对抗样本生成方法[J]. 计算机科学, 2023, 50(7): 325-331.

LI Kun, GUO Wei, ZHANG Fan, DU Jiayu, YANG Meiyue. Adversarial Malware Generation Method Based on Genetic Algorithm [J]. Computer Science, 2023, 50(7): 325-331.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220800176

基于遗传算法的恶意软件对抗样本生成方法

李 坤[1] 郭 威[1] 张 帆[1] 杜加玉[2] 杨梅樾[2]
1 信息工程大学信息技术研究所 郑州 450002
2 紫金山实验室 南京 211111
(moyue_lk@foxmail.com)

摘 要 近年来,随着互联网技术的发展,恶意软件成为网络攻击的重要手段。为防御恶意软件攻击,可以将深度学习技术应用于恶意软件检测。然而,由于深度学习模型自身的局限性,基于深度学习的恶意软件检测模型容易受到恶意软件对抗样本的攻击,导致恶意软件对抗样本逃逸模型检测。通过研究恶意软件对抗样本的生成,可以帮助模型设计者改进模型设计、提升模型鲁棒性和防御能力。因此,针对基于灰度图的恶意软件检测模型,提出一种基于遗传算法的恶意软件对抗样本生成方法。该方法通过遗传算法优化扰动,再结合混淆操作向恶意软件中注入扰动,从而保证生成的恶意软件对抗样本具有对抗性、可执行性和恶意性。经实验验证,相比现有工作,所提方法生成的对抗样本攻击成功率平均提高56.4%。

关键词:对抗样本;深度学习;恶意检测;对抗攻击;遗传算法
中图法分类号 TP309.5

Adversarial Malware Generation Method Based on Genetic Algorithm

LI Kun[1],GUO Wei[1],ZHANG Fan[1],DU Jiayu[2] and YANG Meiyue[2]
1 Institute of Information Technology,University of Information Engineering,Zhengzhou 450002,China
2 Purple Mountain Laboratories,Nanjing 211111,China

Abstract In recent years,with the development of Internet technology,malware has become an important method of network attack. To defend against malware attacks,deep learning techniques can be applied to malware detection. However,due to the limitations of deep learning models,malware detection models based on deep learning are vulnerable to adversarial malware,which leads to adversarial malware evading model detection. By studying the generation of adversarial malware,it can help model designers to improve model design,improve model robustness and defense capabilities. Therefore,for the malware detection model based on grayscale image,the adversarial malware generation method based on genetic algorithm is proposed. It optimizes the perturbation by genetic algorithm,and then injects the perturbation into the malware by the obfuscation operation,so as to ensure that the generated adversarial malware samples are adversarial,executable and malicious. It is verified by experiments that the attack success rate of adversarial samples generated by the proposed method increases by 56.4% on average compared to the existing work.

Keywords Adversarial examples,Deep learning,Malware detection,Adversarial attacks,Genetic algorithms

1 引言

随着时代的发展,计算机系统在人类的生产活动中得到广泛应用。但网络空间中存在大量的恶意软件,这些恶意软件直接威胁到计算机系统的安全,给使用者带来巨大的经济损失。随着深度学习技术的发展,其在图像目标检测[1]和自然语言处理[2]领域得到广泛应用。因此,研究者们开始结合深度学习技术对恶意软件进行检测。

根据检测过程中是否执行恶意软件,可以将基于深度学习的恶意软件检测分为静态检测和动态检测[3]。基于灰度图的恶意软件检测是一种典型的静态检测方法,该方法具有检测便捷、成本低等优点。Cui 等[4]在恶意软件灰度图数据集上,使用基础的卷积神经网络(Convolutional Neural Network,CNN)模型实现恶意软件分类,可以达到94.5%的检测正确率。

然而,由于深度学习模型天然存在鲁棒性不足[5]的缺点,导致基于灰度图的恶意软件检测模型容易受到恶意软件对抗样本的攻击。恶意软件对抗样本由恶意软件添加扰动生成,其可以欺骗检测模型,实现检测逃逸。Grosse 等[6]基于DREBIN 数据集,对恶意软件检测模型展开对抗攻击,令生成的恶意软件对抗样本被判断为良性,从而逃逸模型的检测。

通过研究恶意软件对抗样本生成,可以测试模型的可靠性,也可以促使模型设计者提高模型的对抗防御能力。目前,面向恶意软件灰度图检测的对抗样本生成研究存在恶意软件

到稿日期:2022-08-17 返修日期:2022-11-26
通信作者:张帆(13838267352@qq.com)

对抗样本的可执行性与恶意性无法保持[7]、恶意软件对抗样本的对抗攻击成功率不高[8]、恶意软件对抗样本中扰动注入比例过大[9]等问题。

针对上述问题，本文提出了一种基于遗传算法的恶意软件对抗样本生成方法。该方法可以保持恶意软件对抗样本的可执行性和恶意性，实现较高的对抗攻击成功率，且注入扰动比例在10%以内。本文的主要工作如下：

(1)结合恶意软件混淆操作注入扰动，将扰动注入到不被执行的区域，保证了恶意软件对抗样本与原始样本功能一致，使其具有可执行性和恶意性。

(2)通过遗传算法重复优化注入的扰动，增强扰动的对抗性，以小扰动量实现较高的对抗攻击成功率。

(3)通过向PE文件中直接注入扰动来保证扰动在恶意软件转灰度图的过程中不丧失对抗性，实现恶意软件对抗样本的端到端生成。

2 研究背景和相关工作

2.1 基于灰度图的恶意软件检测

基于灰度图的恶意软件检测是一种具有代表性的恶意软件静态检测方法，其具体流程如图1所示。首先将PE二进制文件转化为灰度图像，本文使用Malimg[10]数据集的规则进行转化。将二进制文件中的8位无符号二进制数转为0-255的十进制数，并将十进制数组合成灰度图图像，图像的宽度与高度设定遵守Malimg[10]数据集规则。最后，将经过尺寸调整的灰度图输入卷积神经网络，实现恶意软件检测分类。

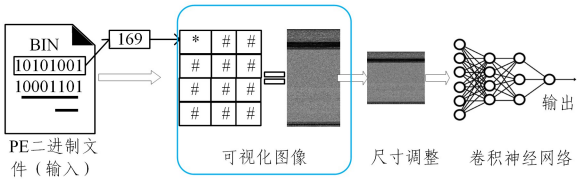


图1 基于灰度图的恶意软件检测流程

Fig. 1 Malware detection process based on grayscale image

2.2 面向恶意软件灰度图检测的对抗样本生成

目前，针对基于灰度图的恶意软件模型，研究者们提出了多种恶意软件对抗样本生成方法。Liu等[11]提出了ATMPA(Adversarial Texture Malware Perturbation Attack)方法，使用FGSM[12](Fast Gradient Sign Method)和C&W[13](Carlini and Wagner)生成扰动，并将扰动添加到PE文件灰度图上，生成对抗样本。Chen等[7]则提出利用生成对抗网络[14](Generative Adversarial Networks，GAN)实现恶意软件对抗样本生成，该方法通过GAN将噪声映射为扰动，并将扰动添加到恶意软件灰度图像上，生成对抗样本。但上述方法生成的对抗样本均为恶意软件灰度图，即使将灰度图转回PE文件，也不具有可执行性。

为保证恶意软件对抗样本的可执行性与恶意性，Khormali等[9]提出了一种改进方法，将ATMPA[11]生成的对抗样本或良性软件附加在原始样本末尾来生成恶意软件对抗样本。该方法生成的对抗样本具有可执行性和恶意性，但注入的扰动量过大，最低注入比例为100%。Benkraouda等[15]则结合C&W[13]方法确定扰动的注入位置，并向注入位置的汇编代码中插入汇编空指令(NOP指令)，以生成对抗样本。该方法生成的对抗样本与原始样本差异小，但搜索注入位置，需要专家知识且时间成本高。

在保证恶意软件对抗样本可执行性与恶意性的前提下，为减少扰动注入量，Xiao等[8]提出了可保留恶意软件对抗样本可执行性和恶意性的合法操作方法(Bytecode Attack Remained Availability and Functionality，BARAF)。该方法通过向PE文件未使用部分注入少量0×00，0×80，0×FF等值的字节，在保持可执行性和恶意性的前提下，破坏恶意软件灰度图的纹理特征，实现对抗样本生成。该方法通用性高，但生成的对抗样本攻击成功率不高。

综上所述，目前针对恶意软件灰度图检测的对抗样本生成研究仍存在局限性：1)实现恶意软件对抗样本的可执行性和恶意性保持时，需要的时间成本过高；2)实现较高对抗攻击成功率时，使用的扰动注入量过大；3)没有直接生成字节形式的扰动，导致扰动经过可视化过程后丧失部分对抗性。本文针对上述问题，提出了一种基于遗传算法的恶意软件对抗样本生成方法，通过结合混淆操作注入扰动，保持了恶意软件对抗样本的可执行性和恶意性；利用遗传算法优化注入的扰动，实现以小扰动量生成有较高对抗攻击成功率的对抗样本；直接生成字节扰动，避免转化过程带来的对抗性损失，实现了恶意软件对抗样本的端到端生成。

3 基于遗传算法生成对抗样本

3.1 对抗攻击的数学定义

首先，本文的被攻击模型为基于灰度图的恶意软件检测模型，因此可以将对抗攻击描述为式(1)－式(4)，其中恶意软件输入为x，检测模型为f，则检测分数为$f(x)$，检测结果为y，以0.5为分类阈值，$f(x)$与y的对应关系如式(1)所示。$y=0$表示输入软件被检测为良性软件，反之，表示其被检测为恶意软件。添加的初始扰动为ε，恶意软件混淆操作为H，遗传算法为G，通过混淆操作将扰动注入恶意软件，生成恶意软件对抗样本(x')(见式(2))。其中，$G(\varepsilon)$为经过遗传算法迭代的最终扰动。当恶意软件被检测为恶意(见式(3))，但生成的恶意软件对抗样本被检测为良性(见式(4))，则表明成功生成对抗样本，实现了对抗攻击。

$$y=\begin{cases}0, & f(x)<0.5\\ 1, & f(x)\geqslant 0.5\end{cases} \tag{1}$$

$$x'=H(x+G(\varepsilon)) \tag{2}$$

$$f(x)\geqslant 0.5 \tag{3}$$

$$f(x')=f(H(x+G(\varepsilon)))<0.5 \tag{4}$$

3.2 基于遗传算法的对抗样本端到端生成方法

根据3.1节的定义，本文恶意软件对抗样本生成可描述为在给定扰动量的约束下，求解最佳的扰动，使生成的恶意软件对抗样本能够逃逸模型检测，故对抗样本生成问题为典型的最优化问题。遗传算法是一种启发式优化算法，该算法通过模拟自然选择中的变异、交叉、选择等操作实现最优化问题的求解。另外，遗传算法可以根据给定的适应度值自动调整求解方向，且更适用于大规模变量的求解，故本文使用遗传

算法解决对抗样本生成问题。

现有的部分工作生成的扰动并非为字节形式，后续需要进行转换操作，该过程不但消耗时间，还可能导致扰动丧失对抗性。本文直接在输入端进行扰动字节的优化和注入，以保证恶意软件对抗样本经可视化过程后不丧失对抗性，实现端到端生成恶意软件对抗样本，具体过程如图 2 所示。

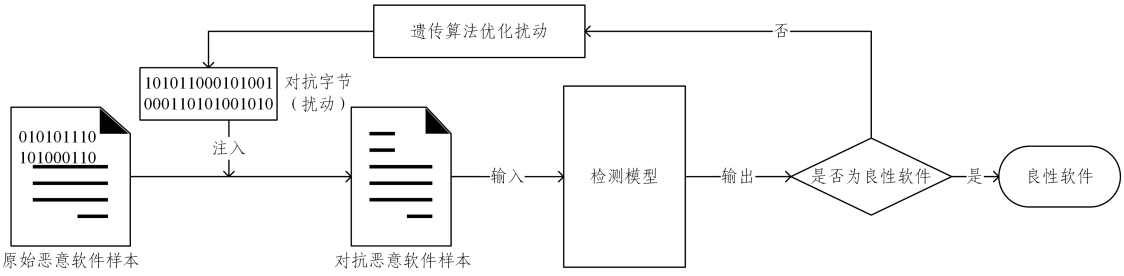


图 2　对抗样本生成流程

Fig. 2　Adversarial example generation process

首先使用混淆操作注入初始扰动 ε，生成恶意软件对抗样本，之后将其通过检测模型，若输出为良性则表明逃逸了检测；反之，则使用遗传算法优化扰动，并将优化后的扰动再次注入原始样本，生成对抗样本。

根据遗传算法的定义，遗传算法优化扰动的过程如算法 1 所示。

算法 1　遗传算法优化扰动

Input：原始恶意软件 x，良性字节 s，恶意软件检测模型 f，恶意软件混淆操作 H，扰动注入比例 rate，种群大小 k，遗传算法最大迭代次数 maxnum

Output：恶意软件对抗样本 x′，遗传算法迭代次数 num

1. size=getsize(x)
2. while i<k−1 do
3.　　a_i←random(size * rate,[0,255])
4. a_{k-1}←s，num←0
5. pop_a←a_i(i=0,1,…,k−1)
6. while i<maxnum do
7.　　$pop_{x'}$←H_{inject}(x,pop_a)
8.　　score←f($pop_{x'}$)
9.　　if score<0.5 break
10.　　mutation(pop_a)
11.　　crossover(pop_a)
12.　　select pop_a with score，num← num+1
13. return x′，num

首先根据输入的种群大小和扰动注入比例，利用随机方式初始化扰动种群，并在扰动种群中加入良性字节扰动个体，该操作的目的是加速遗传算法优化过程（算法第 1－5 行）。之后，开始遗传算法迭代。通过混淆操作将扰动注入恶意软件，生成恶意软件对抗样本，并将其通过检测模型，得到恶意评分，并将其作为遗传算法的适应度值。判断恶意评分是否小于 0.5，若满足则表明成功生成对抗样本，停止迭代过程（算法第 6－9 行）。若生成的恶意软件对抗样本无法逃逸模型检测，则需要对扰动进行变异、交叉、筛选等操作，实现扰动优化。本文结合差分进化算子[16]进行上述操作，具体过程如下。

（1）变异操作（算法第 10 行）。种群个体进行变异、交叉等操作时，需将其编码为基因组形式，如式（5）所示。

$$\vec{X}_{i,G}=[x_{1,i,G},x_{2,i,G},\cdots,x_{D,i,G}],i=0,\cdots,k-1 \tag{5}$$

其中，G 为种群代数，D 为编码后的向量维度，k 为种群大小。则可以通过种群中的个体进行差分、缩放、相加得到变异个体，如式（6）所示：

$$\vec{V}_{i,G}=\vec{X}_{r_1^i,G}+F\cdot(\vec{X}_{r_2^i,G}-\vec{X}_{r_3^i},G) \tag{6}$$

其中，F 为缩放因子（变异参数），其范围为[0.4,1]；r_1^i，r_2^i，r_3^i 为[0,k−1]之间的互斥整数，通过随机的方式生成。

（2）交叉操作（算法第 11 行）。得到变异种群后，可将其表示为式（7），之后将其原始种群进行交叉操作，如式（8）所示。

$$\vec{V}_{i,G}=[v_{1,i,G},v_{2,i,G},\cdots,v_{D,i,G}] \tag{7}$$

$$\vec{u}_{j,i,G}=\begin{cases}v_{j,i,G}, & rand(0,1)<CR \text{ or } j=rand[1,D]\\ x_{j,i,G}, & \text{else}\end{cases} \tag{8}$$

其中，$rand(0,1)$为随机生成的(0,1)之间的实数，$rand[1,D]$为随机生成的(1,D)之间的整数。CR 为交叉参数，其范围为[0,1]。

（3）筛选操作（算法第 12 行）。可将得到的交叉种群表示为式（9）。对种群个体进行一对一生存选择，得到下一代种群，如式（10）所示。

$$\vec{U}_{i,G}=[u_{1,i,G},u_{2,i,G},\cdots,u_{D,i,G},] \tag{9}$$

$$\vec{X}_{i,G+1}=\begin{cases}\vec{U}_{i,G}, & f(\vec{U}_{i,G})\leqslant f(\vec{X}_{i,G})\\ \vec{X}_{i,G}, & f(\vec{U}_{i,G})>f(\vec{X}_{i,G})\end{cases} \tag{10}$$

其中，$f(x)$为适应度计算函数，即本文中的恶意软件检测模型输出。

重复上述过程，当恶意评分小于 0.5 时，停止迭代，输出迭代次数与扰动个体。或者当遗传算法达到最大迭代次数时，停止迭代，输出目前的最佳扰动个体。

本文遗传算法通过 Geatpy[17]框架实现，它是一个高效、模块化、耦合度低、面向对象的遗传进化算法框架。另外，在初始种群中加入良性字节（算法第 4 行）的目的是减少算法迭代次数。根据算法定义，若加入的良性字节可以减少恶意评分，则种群向良性字节方向进化，从而减少迭代次数。若加入的良性字节无法减少恶意评分，则在后续的进化中将其淘汰，不影响算法的最终优化结果。

3.3　恶意软件混淆操作

恶意软件对抗样本与图像对抗样本不同，恶意软件具有语义和离散结构，因此样本特征空间中的扰动操作映射到

实际文件上时，将导致恶意软件样本丧失可执行性。为解决该问题，需利用混淆操作注入扰动，保证恶意软件对抗样本的可执行性。

部分混淆操作通过将扰动注入到程序执行不可达的区域，实现恶意软件可执行性和恶意性的保持。PE 文件结构如图 3 所示[18]，包括 DOS 头、PE 头、可选头、节表、节。其中 PE 头包含文件的结构信息、偏移量等，大部分静态特征可从中获取，节包含文件的主要内容。目前，主要的混淆操作如表 1 所列。

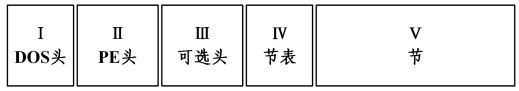


图 3 PE 文件结构

Fig. 3 PE file structure

表 1 混淆操作

Table 1 Obfuscation operations

混淆操作	操作介绍	特点
注入节[19](Inject Section, IS)	改写节的偏移量，在节表(IV)与节(V)之间开辟字节块，并注入扰动	注入空间大小可更改；操作侵入性大
添加字节[20](Padding, PD)	在文件末尾添加字节，注入扰动	注入空间大小可更改；操作侵入性小，操作简单
扩展 DOS 头[19](Extended DOS header, ED)	改写 PE 头的偏移量，扩展 DOS 头区域，并在扩展区域注入扰动	注入空间大小可更改；操作侵入性大；但只适用于程序加载只读取 DOS 头标志位的情况，若程序读取 DOS 头其他内容，会导致程序无法运行
等价替换[21](Equivalent Substitution, ES)	用语义相同的指令替换原始指令，实现局部的扰动的添加	注入空间大小不可更改；操作侵入性较大；成本较高，需要较高的专家知识

本文选取多个混淆操作的目的是将扰动注入 PE 文件的不同位置。其中，等价替换成本过高，且不能实现注入操作，不符合要求。注入节与扩展 DOS 头注入位置相近，均在 PE 文件的头部，但扩展 DOS 头要考虑程序运行时的加载方式。添加字节注入位置为 PE 文件的尾部。综上所述，为使得混淆注入位置分散，操作便捷，本文选取注入节(IS)与添加字节(PD)作为扰动注入方式。

4 实验验证

为验证本文对抗样本生成方法的有效性，评估该方法生成的对抗样本质量、对抗样本生成效果、对抗样本生成影响因素，本文设立了多个实验对照组。实验软件环境使用 Python、PyTorch 框架，硬件环境为配备 NVIDIA T4 GPU 的 Centos7。

4.1 实验参数

本文的实验参数主要有遗传算法参数与注入参数，具体值如表 2 所列。

表 2 实验参数

Table 2 Experimental parameters

种群大小	50
最大迭代	500
交叉参数	0.7
变异参数	0.8
扰动注入方式	IS/PD
扰动注入比例	5%, 8%, 10%

其中遗传算法种群大小会影响求解效率，增加种群数量可以减少迭代次数，但会带来更多的性能开销。交叉参数与变异参数则是根据实验择优选取。注入比例过低会导致扰动量无法在恶意软件可视化图像中形成纹理扰动，故最低扰动比例为 5%。

4.2 数据集及被攻击模型

目前无开源可用数据集，本文数据集来源于网络，其中恶意软件样本来自 VirusShare[22]，共 9 300 个；良性软件样本来自 Windows 系统，共 8 900 个。从恶意软件和良性软件中各选取 1 000 个数据作为测试集，其余为训练集。

另外，目前无开源的恶意软件灰度图检测模型权重，需使用本文数据集训练模型。本文选择两个恶意软件检测模型作为攻击目标，浅层网络为“2C3D”结构，具体网络结构参考文献[9]的描述，结构如图 4 所示，其中激活函数为 ReLU，卷积 filter 大小为(5, 5)，池化 filter 大小为(2, 2)。深层网络为 DenseNet121，具体结构如文献[23]所述。

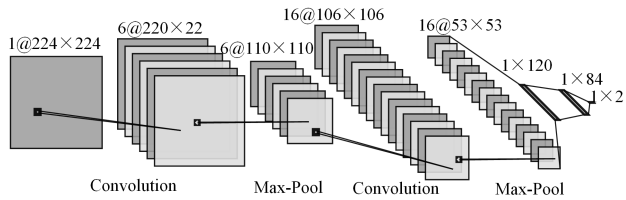


图 4 基于灰度图的恶意软件检测模型结构(2C3D)

Fig. 4 Structure of malware detection model based on grayscale image(2C3D)

选取的被攻击模型均经过研究论证，可以实现恶意软件检测，其余分类网络因未经过研究论证，故未选取。模型训练时，使用相同训练集数据训练模型，测试集用于检验模型泛化能力。两个模型使用动态学习率训练，初始学习率为 0.1，递减倍率为 0.6，步长为 5，经过 50 轮训练，模型测试结果如表 3 所列。

表 3 模型测试结果

Table 3 Model testing results

(单位：%)

模型	*Accuracy*	*auc*
2C3D	98.6	99.86
DensNet121	99.9	99.92

4.3 实验分析

实验从数据集中随机选取 500 个恶意软件样本作为原始样本，进行后续实验操作。原始样本均能被模型检测，模型误分类率(Misclassification Rate, MR)为 0。其余实验参数细节见 4.1 节。

4.3.1　对抗样本质量

恶意软件对抗样本的质量可以通过对抗性、可执行性和恶意性、扰动量进行评估。对抗性评价对抗样本能否欺骗检测模型，若恶意软件对抗样本可以被模型分类为良性，则表明其具有对抗性。当恶意软件对抗样本可以执行且保留了恶意功能，则表明其具有可执行性和恶意性。扰动量表示对抗样本中的扰动量大小，其可以使用扰动添加空间占据原始文件的比例来衡量，最低扰动比例表示该方法生成对抗样本需要的最小扰动量。现有工作生成对抗样本的质量对比如表4所列。其中对抗性指标表示是否具有对抗性，不表示对抗攻击成功率的高低。可执行性和恶意性的验证结果在4.3.2节给出。

表4　对抗样本质量对比

Table 4　Comparison of adversarial samples quality

工作	对抗性	可执行性和恶意性	最低扰动比例/%
ATMPA[11]	√	×	100
Khormali[9]	√	√	100
GAN[7]	√	×	100
BARAF[8]	√	√	5
Our work	√	√	5

4.3.2　恶意软件对抗样本可执行性和恶意性验证

为验证恶意软件对抗样本的可执行性和恶意性保持情况，本文结合LIEF库模拟Windows程序加载读取过程，将恶意软件与恶意软件对抗样本的加载读取内容进行比较，根据内容是否一致，计算该方法生成恶意软件对抗样本的保持率(可执行性和恶意性)，实验结果如表5所列。实验结果表明，通过混淆操作注入节(IS)和添加字节(PD)注入扰动生成的恶意软件对抗样本可执行性和恶意性得到保留。

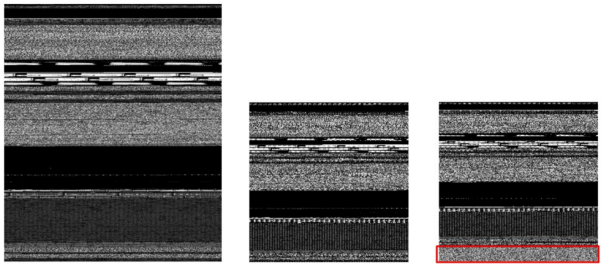
表5　可执行性和恶意性验证结果

Table 5　Verification results of executability and maliciousness

混淆操作	可执行性和恶意性保持率/%
注入节(IS)	100
添加字节(PD)	100

4.3.3　对抗样本生成效果

通过对生成的恶意软件对抗样本进行可视化，可以直观观察到注入的扰动。以添加字节为注入方式，恶意软件对抗样本可视化结果如图5所示。



(a)原始样本　(b)原始样本(尺寸调整)　(c)对抗样本(尺寸调整)

图5　软件样本灰度图

Fig. 5　Software sample grayscale

图5(a)为原始样本的灰度像，图5(b)为图5(a)尺寸调整后的灰度图，图5(c)为对抗样本尺寸调整后的灰度图，图5(c)中框选部分即为注入的扰动。图中结果显示注入的扰动并未导致恶意软件灰度图的纹理特征发生大的变化，表明本文生成的对抗样本并非通过破坏灰度图的纹理特征实现对抗攻击。

对抗样本生成效果可以通过对抗样本的攻击成功率来量化。根据对抗样本定义，当对抗样本被错误分类时，表明对抗攻击成功，则使用误分类率(Misclassification Rate，MR)评估对抗样本生成效果。根据对抗样本质量评估结果，选取BARAF[8]作为对比方法。扰动注入比例设为0.05。BARAF[8]的混淆操作注入空闲字节(Inject Free Bytes，IFB)与本文的注入节(IS)存在区别，注入空闲字节(IFB)指向PE文件中存在空闲字节的位置注入扰动，而注入节是直接注入1节扰动字节。

实验结果如表6所列，其中$MR(0)$是指迭代次数为0时，模型的误分类率(MR)。BARAF[8]无法迭代，故无$MR(500)$的实验数据。$MR(0)$的结果表明本文注入的初始扰动与BARAF[8]相比无明显优势。但$MR(500)$的结果表明初始扰动经过遗传算法优化，生成的对抗样本可以实现较高的对抗成功率。以$MR(500)$下不同混淆操作的成功率平均值作为本文的对抗样本攻击成功率，可以发现，与BARAF[8]相比，本文方法的对抗攻击成功率平均提高56.4%。

表6　对抗样本生成效果

Table 6　Adversarial samples generation effect

对抗样本生成方法	被攻击模型	混淆操作	$MR(0)$	$MR(500)$
BARAF[8]	2C3D	PD	28.6%	无法迭代
		IFB	31.2%	无法迭代
	DenseNet121	PD	3.8%	无法迭代
		IFB	8.6%	无法迭代
Our work	2C3D	PD	36.6%	62.4%
		IS	32.2%	99.6%
	DenseNet121	PD	6.2%	42.8%
		IS	12.8%	93%

本文通过遗传算法增强扰动的对抗性，并注入扰动生成对抗样本，不同对抗样本生成效果所需要的迭代次数不同，故以遗传算法迭代次数为指标，评估本文方法生成对抗样本的性能成本。实验时，扰动注入比例为5%，注入方式为注入节(IS)。当对抗样本的误分类率达到90%以上时，统计遗传算法的迭代次数。不同恶意软件生成对抗样本的遗传算法迭代次数不同，故使用迭代次数的平均值与中位数描述整体的迭代情况。实验结果如表7所列，当扰动注入比例为5%，注入方式为注入节时，通过遗传算法若干次的迭代，生成的对抗样本可以实现90%以上的对抗攻击成功率。

表7　遗传算法迭代次数($MR>90\%$)

Table 7　Genetic algorithm iteration times($MR>90\%$)

被攻击模型	平均数	中位数
2C3D	12	4
DenseNet121	41	33

4.3.4　对抗样本生成的影响因素

为进一步探究生成对抗样本的影响因素，本文排除了遗传算法的参数，则影响因素可分为注入方式和注入比例。当注入方式为添加字节(PD)时，0.05，0.08，0.1扰动注入比例

下，被攻击模型正确率下降情况如图 6 所示(横坐标为遗传算法迭代次数)。结果表明，相同迭代次数下，增加扰动注入量可以提高对抗样本的攻击成功率。而且增加扰动注入量可以提高对抗攻击成功率的上限，但提升幅度有限。

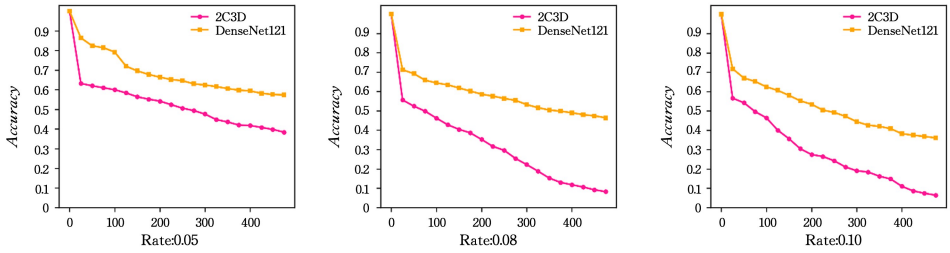


图 6 模型正确率(PD)

Fig. 6 Model accuracy(PD)

当注入操作为注入节(IS)时，0.05，0.08，0.1 扰动注入比例下，被攻击模型正确率下降情况如图 7 所示(横坐标为遗传算法迭代次数)。结果表明在注入节(IS)操作下，增加扰动注入量可以提高对抗攻击率，但对抗攻击成功率的进一步提高需要遗传算法优化的加持，且扰动量的边际效应随着正确率的提高而减弱。

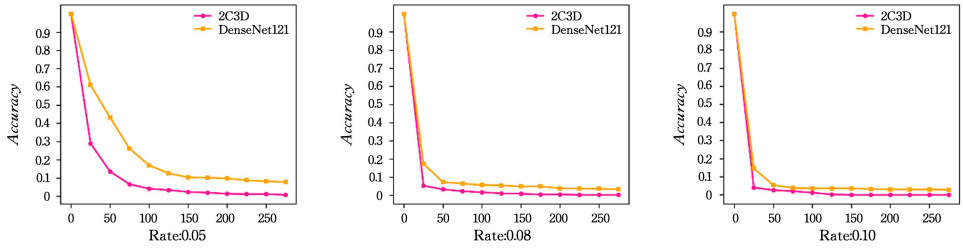


图 7 模型正确率(IS)

Fig. 7 Model accuracy(IS)

4.4 实验总结

经上述实验验证，本文方法生成的恶意软件对抗样本具有可执行性和恶意性，且可以实现较高的对抗攻击成功率。除此之外，图 6 与图 7 的实验结果表明：1)扰动注入量与对抗攻击成功率成正相关；2)不同的注入方式的注入位置不同，导致对抗攻击成功率不同，实际应用中，应根据攻击的目标模型，选取适用的注入方式；3)两个被攻击模型面临相同强度的对抗样本攻击时，模型的正确率下降情况不同，表明可以通过改进被攻击模型结构来提高其对抗防御能力。

结束语 目前，面向恶意软件灰度图检测的对抗样本生成研究中，存在保持可执行性和恶意性、减少扰动注入量、提高对抗攻击成功率等未完成的工作。因此，本文提出了一种基于遗传算法的恶意软件对抗样本生成方法。经实验验证，本文方法生成的恶意软件对抗样本具有可执行性和恶意性，能够以小扰动注入量实现较高的对抗攻击成功率。但本文还有一些工作未完善，首先是扩增扰动注入方式，不同模型对恶意软件不同位置的敏感度不同，因此需要增加扰动注入方式，使扰动注入位置多样化，提高对抗样本的生成效率。其次是实现扰动注入量自适应调整，部分恶意软件较容易生成对抗样本，并不需要过多扰动注入量，故需要针对不同恶意软件选择不同的扰动注入量。

参 考 文 献

[1] GIRSHICK R, DONAHUE J, DARRELL T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 580-587.

[2] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[3] WANG J, ZHANG C, QI X, et al. A Survey of Intelligent Malware Detection on Windows Platform [J]. Journal of Computer Research and Development, 2021, 58(5): 977-994.

[4] CUI Z, XUE F, CAI X, et al. Detection of malicious code variants based on deep learning[J]. IEEE Transactions on Industrial Informatics, 2018, 14(7): 3187-3196.

[5] SZEGEDY C, ZAREMBA W, SUTSKEVER I, et al. Intriguing properties of neural networks[J]. arXiv: 1312.6199, 2013.

[6] GROSSE K, PAPERNOT N, MANOHARAN P, et al. Adversarial perturbations against deep neural networks for malware classification[J]. arXiv: 1606.04435, 2016.

[7] CHEN J, ZOU J, YUAN J, et al. Black-box Adversarial Attack Method Towards Malware Detection [J]. Computer Science, 2021, 48(5): 60-67.

[8] XIAO M, GUO C, SHEN G, et al. Adversarial Example Remaining Availability and Functionality [J]. Journal of Frontiers of Computer Science and Technology, 2022, 16(10): 2286-2297.

[9] KHORMALI A, ABUSNAINA A, CHEN S, et al. COPYCAT, practical adversarial attackson visualization-based malware detection[J]. arXiv: 1909.09735, 2019.

[10] NATARAJ L, KARTHIKEYAN S, JACOB G, et al. Malware images, visualization and automatic classification[C]// Proceedings of the 8th International Symposium on Visualization for Cyber Security. 2011: 1-7.

[11] LIU X, ZHANG J, LIN Y, et al. ATMPA, attacking machine learning-based malware visualization detection methods via ad-

versarial examples[C]// 2019 IEEE/ACM 27th International Symposium on Quality of Service(IWQoS). IEEE，2019：1-10.
[12] GOODFELLOW I J，SHLENS J，SZEGEDY C. Explaining and harnessing adversarial examples[J]. arXiv：1412. 6572，2014.
[13] CARLINI N，WAGNER D. Towards evaluating the robustness of neural networks[C]// 2017 IEEE Symposium on Security and Privacy(SP). IEEE，2017：39-57.
[14] GOODFELLOW I，POUGET-ABADIE J，MIRZA M，et al. Generative adversarial networks[J]. Communications of the ACM，2020，63(11)：139-144.
[15] BENKRAOUDA H，QIAN J，TRAN H Q，et al. Attacks on Visualization-Based Malware Detection，Balancing Effectiveness and Executability[C]// International Workshop on Deployable Machine Learning for Security Defense. Cham：Springer，2021：107-131.
[16] DAS S，SUGANTHAN P N. Differential evolution，A survey of the state-of-the-art [J]. IEEE Transactions on Evolutionary Computation，2010，15(1)：4-31.
[17] geatpy. The genetic and evolutionary algorithm toolbox with high performance in python [EB/OL]. http：//www. geatpy. com/.
[18] Microsoft Inc. PE Format [EB/OL]. https：//docs. microsoft. com/en-us/windows/win32/debug/pe-format.
[19] DEMETRIO L，COULL S E，BIGGIO B，et al. Adversarial exemples，A survey and experimental evaluation of practical attacks on machine learning for windows malware detection[J]. ACM Transactions on Privacy and Security (TOPS)，2021，24(4)：1-31.
[20] KOLOSNJAJI B，DEMONTIS A，BIGGIO B，et al. Adversarial malware binaries，Evading deep learning for malware detection in executables[C]// 2018 26th European Signal Processing Conference(EUSIPCO). IEEE，2018：533-537.
[21] WENZL M，MERZDOVNIK G，ULLRICH J，et al. From hack to elaborate technique—a survey on binary rewriting[J]. ACM Computing Surveys(CSUR)，2019，52(3)：1-37.
[22] VirusShare. com—Because Sharing is Caring [EB/OL]. https：//virusshare. com/.
[23] TEKEREK A，YAPICI M M. A novel malware classification and augmentation model based on convolutional neural network[J]. Computers & Security，2022，112，102515.

LI Kun，born in 1998，postgraduate. His main research interests include artificial intelligence security，adversarial samples，and malware detection.

ZHANG Fan，born in 1981，Ph.D，associate researcher，master tutor. His main research interests include active defense，chip design technology，and high-performance computing.

（责任编辑：何杨）