基于显著性分析的恶意代码对抗样本生成方法

詹达之[1], 孙 毅[2], 张 磊[3], 刘 鑫[1], 郭世泽[1], 潘志松[1]

[1]陆军工程大学 南京 中国 210001
[2]国防科技大学第六十三研究所 南京 中国 210000
[3]军事科学院 北京 中国 100091

摘要 借助于人工智能技术的快速发展, 深度学习模型越来越多得应用于恶意代码检测。由于深度学习模型具有更好的泛化性能, 使其可以处理新的、未知的恶意代码, 能够更好地应对日益增长的恶意代码威胁。然而, 深度学习模型容易收到对抗样本的欺骗, 即攻击者通过对样本进行微小的改动使模型预测错误。该脆弱性带来潜在的安全风险, 导致恶意代码检测系统的鲁棒性大大降低。研究深度学习模型与对抗样本之间的对抗机理, 利用生成的对抗样本挖掘恶意代码检测模型的弱点, 增强模型分类的可解释性是评估和提高恶意代码检测系统鲁棒性的关键。因此, 本文提出一种基于显著性分析的恶意代码对抗样本生成方法, 首先使用可解释性技术分析模型检测恶意代码时输入特征的显著值分布情况, 并对深度学习模型分类恶意代码的决策进行解释。然后挖掘 PE 文件中适合施加对抗扰动的非执行区域字节序列, 并构建了基于显著性分析的恶意代码对抗样本生成框架 SAM。通过修改代码非执行区域中少量的关键字节, 得到功能保留且能有效规避检测的对抗样本。实验结果表明, 本文提出的 SAM 方法在仅修改不超过 1024 个字节的情况下, 生成的对抗样本在白盒模式下对 MalConv 模型实现了 72.9%的规避成功率, 黑盒模式下的成功率也达到了 45%, 相较其他方法有明显提升。

关键词 恶意代码检测; 深度学习; 对抗样本; 显著性分析
中图法分类号 TP309.5 DOI 号 10.19363/J.cnki.cn10-1380/tn.2024.11.05

Generating Adversarial Malware Examples Based on Saliency Analysis

ZHAN Dazhi[1], SUN Yi[2], ZHANG Lei[3], LIU Xin[1], GUO Shize[1], PAN Zhisong[1]

[1] Army Engineering University of PLA, Nanjing 210001, China
[2] The Sixty-third Research Institute, National University of Defense Technology, Nanjing 210000, China
[3] Academy of Military Sciences, Beijing 100091, China

Abstract With the rapid development of artificial intelligence technologies, deep learning models are increasingly being used for malware detection. Deep learning models are better able to deal with the growing threat of malware due to their better generalization performance, which allows them to handle new and unknown malware. However, deep learning models are vulnerable to the adversarial examples, where an adversary makes the model predict incorrectly by making minor changes. This vulnerability poses a potential security risk and leads to a significant reduction in the robustness of malware detection systems. Studying the adversarial mechanism between deep learning models and adversarial examples, mining the weaknesses of malware detection models using the generated adversarial examples, and enhancing the explainability of model classification are the keys to evaluate and improve the robustness of malware detection systems. Therefore, this paper proposes a method for generating adversarial examples of malware based on saliency analysis, which first uses explainable techniques to analyze the distribution of saliency values of input features when the model detects malicious code and to interpret the decision of the deep learning model to classify malicious code. Then, we mine the byte sequences of non-executable regions in PE files that are suitable for applying adversarial perturbations, and construct a generation framework SAM (Saliency-based Adversarial Malware examples), which generates function-preserving and effective adversarial examples that can evade detection by modifying the salient bytes in the non-execution region of the code. The experimental results demonstrate that the SAM proposed achieves a 72.9% evasion rate against the MalConv in white-box mode and 45% in black-box mode with only modifications of no more than 1024 bytes, which is a significant improvement compared to other methods.

Key words malware detection; deep learning; adversarial example; saliency analysis

通讯作者: 潘志松, 教授, Email: panzhisong@aeu.edu.cn。
本课题得到国家自然科学基金(No. 62076251, No. 62106281)资助。
收稿日期: 2023-01-31; 修改日期: 2023-05-11; 定稿日期: 2024-09-05

1 引言

恶意代码是目前网络安全面临的主要威胁之一, 根据卡巴斯基实验室 2020 年底的统计数据, 平均每天检测到 54 万个恶意代码[1], 其中超过 90%是 Windows PE 恶意代码。传统基于签名的检测手段[2]通过将给定的可疑软件的签名与之前收集的恶意代码签名数据库进行比对来确定其恶意性, 往往只能检测已知的恶意代码, 难以应付数量剧增和不断变化的恶意代码威胁[3-4]。深度学习方法以其能够自动提取难以发现的规则和深层次特征, 在图像、文本分类等各类任务中[5]已取得了巨大成功。受此启发, 研究人员提出了各种基于深度学习的恶意代码检测算法来检测 Windows PE 恶意代码[6-11]。然而, 现有研究表明深度学习算法容易被对抗样本所欺骗。不同种类的恶意代码对抗样本生成算法[12-20]已被提出以规避检测。

NVIDIA 公司研究团队提出的 MalConv 模型[6]是一种基于深度学习的恶意软件检测器, 具备高准确度, 可扩展性、开源等特点, 被认为是大多数静态对抗攻击的基线。它可以处理传统检测器难以识别的未知的、未经修改的和高度变形的恶意代码。该模型通过使用卷积神经网络分析二进制文件的内容, 学习深层次的特征, 并将恶意代码的特征与良性代码的特征进行比较, 以便能够识别未知的恶意代码。MalConv 已在多个数据集上进行了测试, 并获得了出色的准确度。作为恶意代码检测的一个有力工具, 其在网络安全领域具有广泛的应用前景。

然而, 深度神经网络是一个黑盒模型, 其由大量的非线性函数组成, 导致难以直观地剖析其内部机制, 对模型得出结果的依据做出解释[21-22]。现有的恶意代码对抗样本生成方法主要专注于提升规避成功率, 却很少解释模型为什么会被对抗样本所欺骗。对可解释的对抗样本生成方法开展研究, 剖析对抗机理, 挖掘模型弱点, 可以进一步提升对抗样本的性能和提升检测器的鲁棒性。

因此, 本文从恶意代码检测模型的可解释性作为切入点, 针对 MalConv 模型[6], 分析检测时恶意代码字节的显著性分布情况, 揭示深度学习模型区分恶意代码和良性代码的关键特征, 进而提出基于显著性分布的恶意代码对抗样本生成方法。在不同的攻击条件下, 通过优先对文件非执行区域的关键字节添加对抗扰动, 在保留 PE 文件原始恶意功能的条件下, 以较小的扰动实现成功规避。本文的三个主要贡献如下:

(1) 使用两种可解释性技术: 积分梯度 (Integrated Gradient) [23]和 Grad-CAM[24] (Gradient-weighted Class Activation Mapping) 对恶意代码检测模型进行了显著性分析, 定位影响模型决策的关键特征。我们对模型的分类归因给出解释, 认为 MalConv 模型进行检测时, 恶意代码二进制文件的文件头和数据段往往被赋予更高的权重。

(2) 剖析 PE 文件结构特性, 挖掘文件中不同部分的非执行区域, 通过修改该区域内字节添加对抗扰动, 实现对抗样本生成过程中不会破坏原始文件的恶意功能。

(3) 基于显著性分析得到的关键字节定位, 进行高细粒度的扰动操作, 针对白盒和黑盒条件, 提出了恶意代码对抗样本生成方法 SAM 和 SAM-GA。在修改不超过 1024 个字节的情况下, SAM 实现了对 MalConv 模型 72.9%的攻击成功率, SAM-GA 也达到了 42%, 在小扰动下实现成功规避, 相较其他方法有明显提升。

本文剩余章节安排如下: 第 2 节简述对抗样本及恶意代码对抗攻击等研究背景和相关工作; 第 3 节介绍 Windows PE 文件中非执行区域; 第 4 节分析恶意代码分类过程中的显著性分布; 第 5 节描述基于显著性分析的恶意代码对抗样本生成方法; 第 6 节进行实验验证和分析; 第 7 节总结本文工作。

2 相关工作

2.1 基于深度学习的恶意代码检测

根据提取特征的种类, 基于深度学习的恶意代码检测方法可主要分为基于静态特征的方法[6,8-9]和基于动态特征的方法[2,7,10-11]。动态特征是指将样本在如沙箱的隔离环境中执行, 获取的样本调用系统资源、文件、注册表、网络信息和其他方面的运行特征; 静态分析是在不执行样本的情况下, 直接从二进制文件中提取重要特征, 例如字节序列、可读字符串、头信息和灰度图像等。由于基于静态特征的检测方法不需要实际执行二进制文件, 开销更小, 所以常常作为系统安全的第一道防护, 本文也主要关注基于静态特征的检测方法。

恶意代码的字节序列长度往往达到百万数量级, 现有网络结构难以直接输入原始二进制序列检测其恶意性。借鉴卷积神经网络的结构, NVIDIA 公司研究团队 Raff 等人[6]提出 MalConv 检测模型, 借助神经网络的表征能力, 提取恶意代码中的潜在信息和空间相关性,直接将 PE 文件的原始字节序列作为输入。该方法不需要手工构建特征或编译器方面的知

识, 对于恶意代码的变种具有普适性和鲁棒性, 同时由于使用浅层神经网络和大卷积核, 计算的复杂程度与序列长度呈线性关系, 使得模型更适用于检测大型文件, 使其成为第一个有能力处理超过 200 万长度的原始字节序列的网络架构。更详细的关于 MalConv 的信息见附录 A。

2.2 对抗样本

对抗样本的概念首先由 Szegedy 等人[25]提出, 是指在原输入样本上添加不易察觉的对抗扰动, 即可误导模型分类错误。假设 f 是目标分类模型, x 是真实标签为 y 的输入样本, 对抗样本 x^{adv} 可定义为

$$\begin{aligned} &f(x^{adv}) = f(x+\delta) \neq y \\ &\text{s.t. } \|\delta\|_p < \varepsilon \end{aligned} \tag{1}$$

其中 δ 是添加的对抗扰动, 约束促使添加的扰动尽量小。根据攻击者对目标模型的了解程度, 对抗样本生成可以进一步分为白盒模式[15,25-29]和黑盒模式[14,18,30-33]。白盒模式是指攻击者掌握目标模型的所有信息(如架构、权重/参数、输出、特征等), 以及训练目标模型的数据集。相比之下, 黑盒模式指的是攻击者只能获得模型的输出, 无法掌握更多的信息。

虽然研究学者已在自然图像、文本等领域提出多种对抗样本生成方法, 但却无法直接用于生成恶意代码的对抗样本[34]。在图像领域, 攻击者可以修改图像中任意的像素, 只要不被人眼发觉。而在恶意代码中, 除了不易发觉的约束外, 随意的修改二进制字节可能导致文件无法执行或者原始的恶意行为被破坏, 即恶意代码的对抗样本需具有功能保留性。因此在生成恶意代码对抗样本的过程中, 需要更巧妙的设计以满足上述约束。

2.3 恶意代码对抗样本生成方法

Anderson 等人[18]提出在恶意代码中添加对抗扰动最直接且简单的办法是修改 PE 文件中非执行区域的部分字节, 或者直接在文件末尾追加字节。这种字节级别的修改操作由于不会破坏恶意代码源文件的关键字节, 所以能保留原始的恶意功能。基于这种扰动方式, 后续研究学者提出了多种方法[12,15-16,19-20], 本文在此基础上拓展并进行改进。

白盒攻击方法: Kreuk 等人[12]提出在 PE 恶意代码的末尾添加对抗扰动。首先在嵌入空间中用基于梯度的方法生成对抗扰动, 然后通过搜索扰动在嵌入空间的最近邻字节值生成恶意代码对抗样本。在此基础上, Suciu 等人[15]认为算法收敛时间随着添加字节数的增加而线性增长, 并提出 FGM 方法来提升恶意代码对抗样本生成的高效性。

Chen 等人[16]认为攻击中添加的对抗字节往往由随机噪声初始化, 这可能导致攻击性能降低。为了解决这个问题, 提出了两种新型的白盒攻击方法 BFA 和 Enhanced-BFA。BFA 根据显著性向量从良性代码中选择数据块, 然后附加到原始 PE 恶意代码的末尾。在此基础上, Enhanced-BFA 使用 FGSM 方法来迭代优化由 BFA 产生的扰动, 获得更高的攻击性能。

Demetrio 等人[19]认为 MalConv 主要是基于从 PE 头学到的特征来区分 PE 恶意代码和良性 PE 文件。受此启发, 他们提出了一个基于梯度的变体白盒攻击, 通过改变 PE 文件头中的部分字节注入对抗扰动。

黑盒攻击方法: 在真实网络攻防场景, 往往无法获得模型的信息, 即需要在黑盒设置下实现对抗攻击。Chen 等人[16]也提出了针对 MalConv 的黑盒对抗攻击方法 Experience-based。首先, 从良性软件中随机选择数据块, 并将其附加到恶意代码中以产生对抗样本。在进行了多次随机攻击后, 根据数据块的成功轨迹经验计算每个数据块的贡献度。最后, 根据数据块的贡献度排序将其附加到恶意代码的末尾。

Demetrio 等人[20]提出了一个针对 PE 恶意代码检测器的通用对抗攻击框架(RAMEn), 该框架使用遗传算法来生成对抗扰动, 然后通过扩展 DOS 头和移动 PE 文件中第一个节的内容来注入对抗扰动。

综上所述, 我们认为目前的恶意代码对抗样本生成技术存在一定的局限性: (1)基于不现实的白盒设定, 需掌握目标模型的特征类型、模型参数等信息; (2)为了满足对抗样本的功能保留性, 攻击性能受到限制; (3)由于尾部的字节显著性不高, 导致现有尾部填充的对抗方法往往需要添加大量的对抗字节。

本文针对上述问题, 我们提出了一种基于显著性分析的恶意代码对抗样本生成方法, 利用显著性分析技术揭示目标模型学习的关键特征, 通过优先修改文件非执行区域内对分类影响大的字节, 在不同攻击条件下生成能规避检测且功能一致的恶意代码对抗样本。

3 PE 文件非执行区域分析

3.1 PE 文件结构

可移植可执行文件(Portable Executable, PE)[35]是微软旗下 Windows 平台主流的可执行文件格式, 包含了拓展名为 EXE、DLL、SYS、COM、OCX 等的多种文件。PE 格式是通用对象文件格式 (Common Object File Format , COFF) 格式的变种, 提供一种通用的文件格式, 以用于执行代码和存储 DOS 存根的

基本数据。文件所包含的内容主要可分为三部分: 文件头信息、节信息和未映射数据。

第一部分是文件头信息, 用于提供信息指导操作系统将 PE 文件映射到内存中执行, 主要包含 DOS 头、NT 头和节表。其中 DOS 头是为了兼容微软桌面操作系统(Microsoft disk operating system, MS-DOS)[36]而存在。NT 头概述了整个 PE 文件的概况信息, 包括节数、创建时间以及 PE 文件的其他各种属性等等。节表提供了所有节的信息, 包括名称、偏移量、大小和其他信息。

节信息中的内容是 PE 文件的基本代码或数据单元。每个节都是一个连续的结构, 每次加载必须连续加载节中的所有原始数据。PE 文件通常包括的节如表 1 所示。

表 1　PE 文件中常见节名及包含内容

Table 1　Common section names and contents of PE files

节名	内容
.text	通常用来存放程序执行代码
.data	通常用来存放程序中已初始化的全局变量
.idata	包含程序所需的 DLL 文件信息
.edata	包含所有提供给其他程序使用的函数和数据
.rsrc	包含程序所需的资源数据
.reloc	如果加载 PE 文件失败, 将基于此节进行重新调整

最后一部分是未映射数据。PE 文件在执行的时候, 映射到内存中的结构布局与该文件在磁盘中存储时的结构布局是一致的。因此, 包括 PE 文件末尾未使用的字节在内的部分数据, 如调试信息, 并没有被映射到内存中。

3.2　可修改字节序列

非执行区域是指 PE 文件中没有实际执行的代码块, 修改这部分代码并不会影响原始软件的功能性。为了实现基于显著值权重优先修改对分类影响大的字节并生成对抗样本, 本节将分析 PE 文件中可添加对抗扰动的区域。结合 PE 文件的构造, 从 3 个部分中挑选出可修改字节序列, 即部分 DOS 头数据, 节空隙和尾部填充数据。

部分 DOS 头数据: DOS 头包含 DOS Header 和 MS-DOS 存根两个结构体。其中 DOS Header 是为了兼容 DOS 程序而设置, 而 MS-DOS 存根是一个在 MS-DOS 下运行的有效应用程序, 当代码在 MS-DOS 中运行时, 输出消息“此程序不能在 DOS 模式下运行”。

DOS Header 共 64 个字节, 仅有两个有意义的字段 e_magic 和 e_lfanew。e_magic 字段被称作幻数, 长度为 2 字节, 用于标识文件类型类型, e_lfanew 字段位于位置 0x3c, 存放有 PE 签名的文件偏移量, 此信息使 Windows 能够正确执行映像文件。如果这两个字段被修改, 程序可能无法正确执行, 然而其余的 58 个字节可以用作注入对抗扰动。

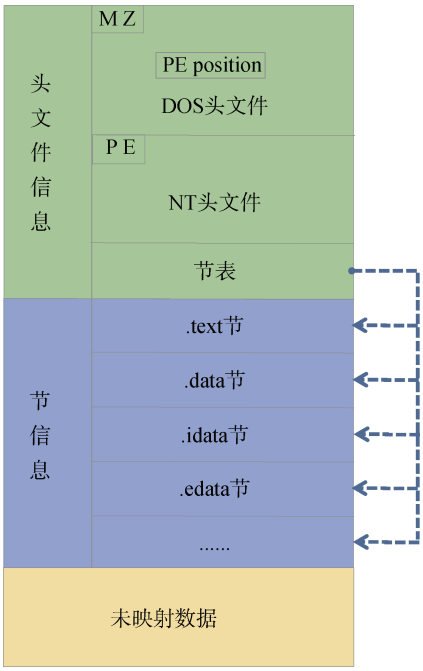


图 1　PE 文件结构

Figure 1　Structure of PE files

节空隙数据: 在 PE 文件中, 每个节都是一个连续的结构, 其大小根据磁盘对齐值进行对齐。每一块区块从对齐值倍数位置开始存放, 对于没有填满的部分会用 0x00 填充, 这些 0x00 字节就构成了节间隙。例如: 对齐值为 200h, 一个区块大小为 50h, 它从 100h 开始存放, 直到 150h 结束, 其后的 150h 到 300h 就用 0x00 填充, 为节空隙。对于每个节, 间隙大小为 $Rawsize - Virtualsize$, 其中 $Rawsize$ 是对齐后的占用空间大小, $Virtualsize$ 是对齐前的实际数据大小。间隙的索引是从 $RawAddress + Virtualsize$ 到 $RawAddress + Rawsize$。由于该区域不包含实际的数据内容且在加载后不会被执行, 所以修改节间隙不会破坏恶意软件的原始功能。

尾部填充数据: 许多恶意代码对抗攻击方法[12,16,19-20]提出在原始恶意代码文件尾部附加字节, 由于填充的数据未在节表中进行定义, 因此不会被链接器映射到内存中, 故不会被实际执行。然而在尾部填充数据存在许多局限, 除了不能向已经超过模型最大输入尺寸(例如 MalConv[6]限制为 2MB)的文

件追加字节外, 现有研究表明对分类产生较大影响的特征(即表现出恶意的特征)往往集中在文件的前部[18-20], 在尾部添加扰动无法有效更改这些特征。因此本文选择在PE文件中的多个区域基于显著性权重添加对抗扰动, 有效更改显著特征, 实现在较小扰动规模的情况下更高的攻击成功率。

综上, 如图2绿色区域所示, 本文定义了可修改字节序列 $idx=[d,s_1,\ldots,s_n,p]$, 共包含 $n+2$ 个连续的字节区域, d 表示部分 DOS 头数据, $s_1,\ldots,s_n$ 表示节空隙数据, 其中 n 是含有空隙的节的总数, p 表示尾部填充数据, 具体步骤见算法1。

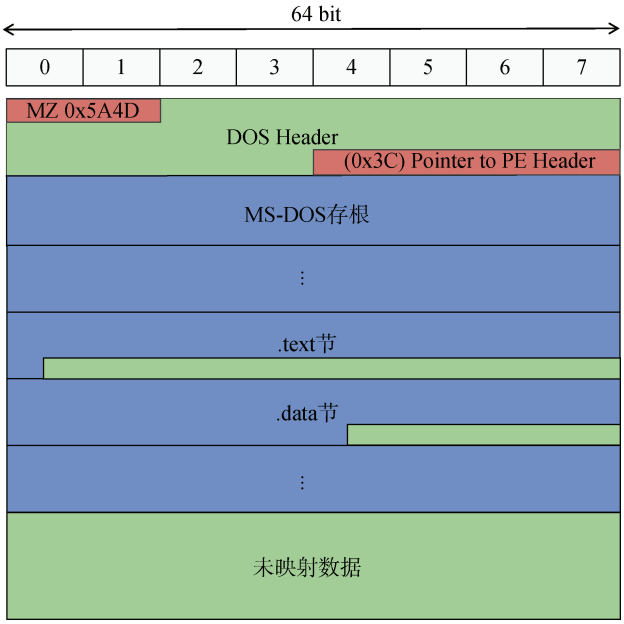


图 2 PE 文件非执行区域示意

Figure 2 Non-executable regions of PE files

算法 1: 定位 PE 文件非执行区域

输入: PE 恶意代码二进制文件 x, 填充字节数 N

输出: 可修改字节索引 idx

1. 部分文件头数据

$PartialDos \leftarrow [2,\ldots,59]$

2. 解析 PE 文件获取节空隙信息

$S \leftarrow GETSECTION(x)$

FOR s_i IN S DO

$s_i.Rawsize \leftarrow r$

$s_i.Virualsize \leftarrow v$

$s_i.Rawaddress \leftarrow add$

3. IF $r > v$ Then

$Slack.APPEND[v+add : r+add]$

End IF

END FOR

4. 在文件尾部填充字节

$Pad \leftarrow [|x|,\ldots,|x|+N]$

5. 合并 3 个区域并返回

$idx \leftarrow PartialDos \bigcup Slack \bigcup Pad$

RETURN: idx.

4 恶意代码显著性分析

本文改进的主要思路是优先修改对分类影响大的字节, 进而实现在较小扰动的情况下成功规避检测器。然而, 深度神经网络的不可解释性导致我们无法直接推断模型作出分类的归因[21]。因此在这一节中, 我们使用积分梯度[23]和 Grad-CAM[24]两种可解释技术对 MalConv 检测模型进行显著性分析, 即定位恶意代码不同字节在分类时“最体现恶意性”的区域, 以便于后续在这些关键区域施加对抗扰动。

积分梯度[23]已使用在解释图像和文本分类等任务上[37-39], 但该方法使用于二进制程序上必须设定一个输入基线值, 基线的输出接近为 0。对于一个图像识别系统来说, 基线是一张全黑的图片, 对于一个自然语言处理系统而言, 基线是全部值为 0 的词向量。对于二进制文件, 存在两个可能的基线选择: (1)空文件, 和(2)零值文件。对于 MalConv, 空文件是一个由特殊的填充数字 256 填充的向量, 由于零值文件会以一定置信度被视为恶意代码, 不满足输出为 0, 所以选择空文件作为基线。

图 3 显示了通过积分梯度分析 MalConv 对输入的恶意代码样本的文件头部分字节的显著值, 用颜色标记每个字节的贡献。红色的单元格象征着 MalConv 认为这些字节具有恶意性, 相反, 蓝色的单元格代表良性。我们注意到该部分字节主要为红色, 说明呈现出恶意性。为了更好得观察文件整体的显著分布情况, 我们在图 4 中展示了模型分配给文件的不同组成部分的显著值情况。直方图的每个条目都是该区域内字节的显著值和, 其颜色描述方式与图 3 相同。

正如文献[18-20]提到的, MalConv 将较高的权重放在文件的开头, 文件头对恶意软件分类具有较大的影响, 即使只有很小占比的字节数量。但是文件头的结构相对固定, 即在良性软件和恶意软件中该区域的部分字节是相同的。但从实验结果可以看到 MalConv 对这些中性的“相同字节”也赋予了权重, 我们猜测一方面是文件头虽然通常不包含实现功能的恶意载荷, 但包含了恶意软件的元数据和属性信

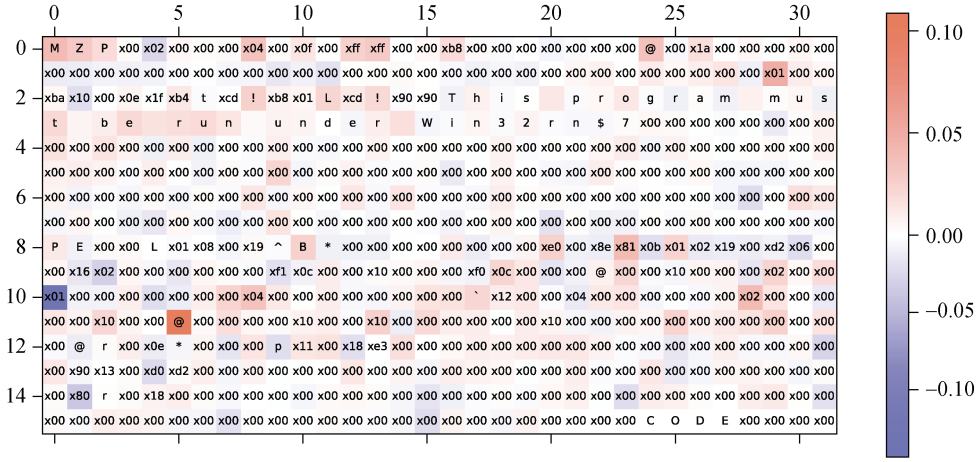


图 3　MalConv 对输入恶意代码样本的文件头字节的显著值分布

Figure 3　MalConv's distribution of saliency values for the headers of the input malware sample

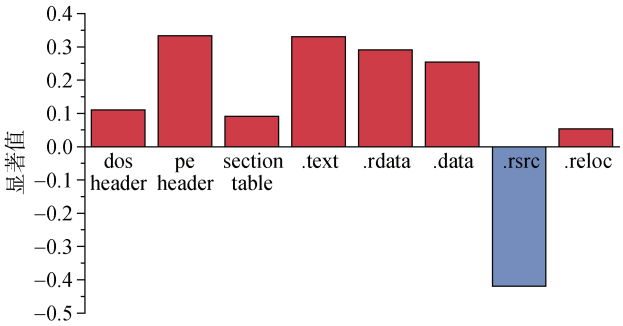


图 4　使用积分梯度分析恶意代码样本中不同区域的恶意程度

Figure 4　Analyze the maliciousness of different regions in malware samples based on integrated gradient technology

息，这些信息被用来欺骗用户和安全软件，以达到隐藏和保护恶意行为的目的。例如被修改的时间戳、标志位，以及对文件头进行编码和加密等，这些属性可能被作为恶意代码的特征。另一方面，模型结构导致可解释技术对特征的定位存在偏差。MalConv 使用了尺寸为 500 的大卷积核和步长，虽然增加了视野域，但使得同一滑窗内的不同类型的字节的属性“难以区分”。

此外，不同节的显著程度也有较大差异，例如存放可执行代码的.text 和数据的.rdata 及.data 被赋予较大的恶意显著值，因为实现恶意功能的代码和数据就可能存放于此，而存放菜单、图标等资源数据的.rsrc 节则被赋予较大的良性显著值。为了提升方法的可信程度，我们也使用 Grad-CAM 技术[24]来获得 MalConv 模型的显著值分布情况。Grad-CAM 是 CAM(Class Activation Mapping)[22]的泛化版本，通过 Grad-CAM 我们能够绘制出模型注意力的热力图，图中分数越高(颜色越“热”)的地方表示在输入图片中这块区域对网络的响应越高、贡献越大，即得到了对于给定类别，模型分类的重点关注区域。该技术将模型最后一个卷积层输出作为激活图，通过加权平均得到卷积特征图尺寸相同的粗略显著值图，然后上采样到输入尺寸。对于 MalConv，通过 Grad-CAM 分析得到的热力图尺寸大小为 4039×1，显著值粒度与卷积核大小一致，为 500。

为了直观的分析结果，我们使用 Grad-CAM 分别在一个恶意软件和良性软件上求取显著值，出于便于可视化的目的，结果被转换为宽度为 500 的彩色图像，颜色方案与前文一致，如图 5 所示。通过比较，可以发现恶意代码的红色(恶意)区域明显比良性区域多。同时恶意软件的头部也展现出明显的恶意性，这与使用积分梯度方法得出的结论一致。关于积分梯度和 Grad-CAM 技术更细节的信息见附录 B。

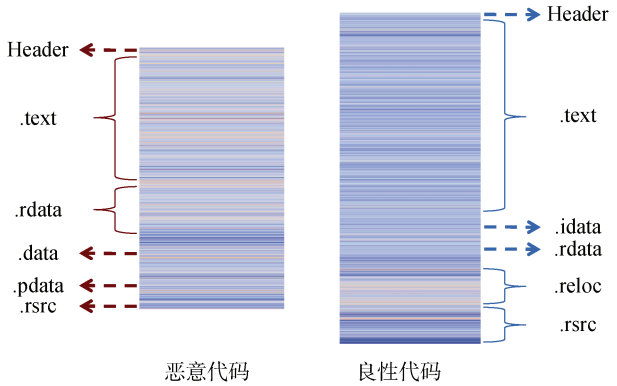


图 5　使用 Grad-CAM 分析恶意代码样本中不同区域的恶意程度

Figure 5　Analyze the maliciousness of different regions in malware samples based on Grad-CAM technology

5 恶意代码对抗样本生成

上节中我们对模型的显著性区域进行了分析, 本节将主要介绍如何基于显著性分析结果生成能规避检测的恶意代码对抗样本。为解决现有方法往往添加大量扰动的缺点, 本文提出根据显著性分析优先修改对分类影响大的字节, 在较小扰动下实现成功规避。根据攻击条件的不同, 在第 5.1 节提出针对白盒设置的 SAM 方法, 同时为了进一步拓展攻击的可行性, 我们也提出了黑盒模式下的恶意代码对抗样本生成方法 SAM-GA, 并在第 5.2 节进行介绍。

5.1 白盒模式 SAM 方法

本文的关键是定位对分类影响大的字节, 在白盒条件下攻击者能够获取模型的结构和内部参数。正如第 4 节所做的工作, 我们可直接使用可解释技术积分梯度和Grad-CAM对目标模型显著性分析以获取字节显著值, 分析定位得到关键字节并添加对抗扰动。恶意代码对抗样本的生成过程如算法 2 所示。

算法 2: SAM 算法

输入: PE 恶意代码二进制文件 x, 目标模型 M, 可修改字节索引 idx, 修改字节数 γ, 迭代次数 T

输出: 恶意代码对抗样本 x^{adv}

1. 显著性分析得到显著值分布 S
   $S \leftarrow SALIENCYANALYSIS(x, M)$
2. $E_i = \hat{\phi}(i)$, $\forall i \in [0,256]$, $x^{adv} \leftarrow x$
   FOR n IN $[0,\ldots,T-1]$ DO
   $Z \leftarrow \phi(x)$, $G \leftarrow -\nabla_{\phi}(Z)$
3. 选择显著值最高的字节进行修改
   FOR j IN argsort $(S,\gamma) \wedge j \in idx$ DO
   $g_j \leftarrow G_j / \left\| G_j \right\|_2$
4. 寻找最近邻字节值
   FOR i IN $[0,\ldots,255]$ DO
   $s_i \leftarrow g_j^T (E_i - Z_j)$
   $d_i \leftarrow \left\| E_i - (Z_j + g_j \cdot s_i) \right\|_2$
   END FOR
   $x_j^{adv} \leftarrow \text{argmin}_{i:s_i>0} d_i$
   END FOR
   END FOR
   RETURN: x^{adv}.

与文献[16]方法类似, 在得到输入样本 x 的显著值分布 S 后, 攻击先对嵌入空间内的字节值进行编码得到其嵌入向量。然后开始迭代, 首先将输入样本 x 映射到嵌入空间 $\phi(\cdot)$ 中, 并计算出梯度的负值 G (步骤 2)。根据显著值分布 S, 从可修改字节索引 idx 中挑选出显著值最高的 γ 个字节, 并对每个字节对应的梯度进行归一化(步骤 3)。对于每个挑选出的字节 j, 计算出一条经过该字节在嵌入空间映射点的直线, 方向是归一化后的梯度方法。在这个方向上投影所有 256 个字节值的嵌入向量, 并计算每个嵌入向量与该线的距离 d_i, 最后将原始字节替换为与最近邻的嵌入向量相对应的字节(步骤 4)。

5.2 黑盒模式 SAM-GA 方法

现有的可解释性技术通常需要根据模型神经元的激活信息或者输出来推断模型的决策, 但在黑盒设定下攻击者无法访问模型的内部参数, 因此很难直接对黑盒模型进行显著性分析。在第 4 节中, 我们已经观察到模型分配给文件的不同组成部分的显著值情况, 对于任意的输入恶意代码, 其显著值分布存在一定的规律。分析认为可以将此规律转换为先验知识, 添加到黑盒攻击的优化迭代过程中, 并以此提出 SAM-GA 算法。

在图 6 中我们统计了在整个数据集上可修改字节序列 $idx = [d, s_1, \ldots, s_n, p]$ 的显著值分布情况, 并挑选了部分节进行展示。直方图的每个条目反映了区域内单个可修改字节的平均显著值, 并按照从大到小排序。从图中我们可以观察到, 除了前文分析的 MalConv 模型认为文件头和数据节含有恶意性外, 存储压缩或加密代码的.packed 节和存储非初始化数据段的.b 节被赋予了一定显著性, 这是由于攻击者往往会使用压缩或加密技术来隐藏程序代码, 以逃避杀毒软件的检测和防御, 也常将恶意软件的加密密钥、配置信息等重要数据放置在.b 节中。同时有趣的是, 模型认为 UPX 加壳产生的 UPX 节以及 FGS 压缩器压缩后产生的.z2 节是良性的, 这暗示对恶意代码使用 UPX 加壳或者特定的压缩算法可能是规避 MalConv 检测的方法。此外, 由于尾部填充字节初始值为基线值 (256), 其显著值为零或者保持较低水平。

本文提出将恶意性平均水平转换为施加对抗扰动时字节修改的顺序, 即显著性先验。方法选择修改字节的步骤如下, 首先优先修改部分文件头中的数据 d, 将 d 添加到选择字节序列 $bytes$; 然后如修改字节总数 γ 大于 $|d|$, 则查询文件的.data 节是否有空隙字节, 如果有就添加到选择字节序列 $bytes$; 以此类推, 直至使 $bytes$ 的字节数达到 γ。选择好修改的

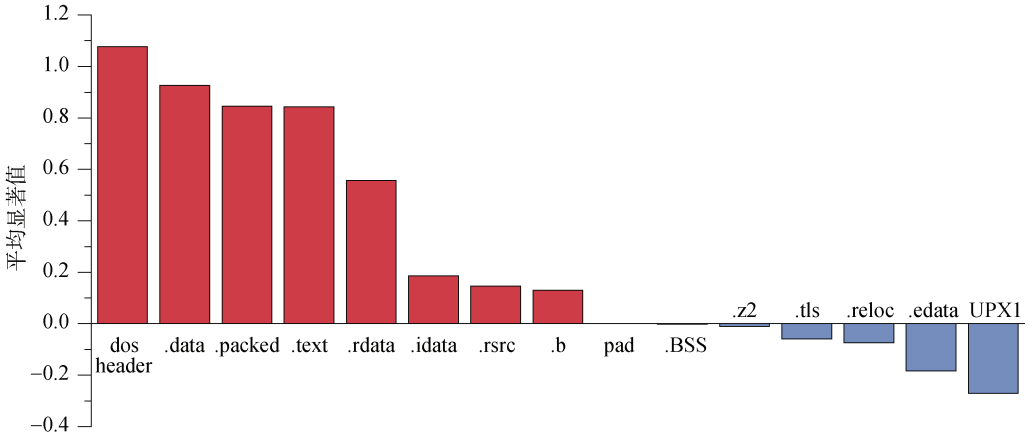


图 6　恶意代码样本中不同区域的恶意性平均水平

Figure 6　Average level of maliciousness in different regions of the malware sample

字节后, 就可以使用遗传算法生成对抗扰动。遗传算法通过一个适应度函数评估个体对环境的适应程度。如果直接使用模型输出的硬标签(0: 良性, 1: 恶意代码)作为适应值, 可能无法提供足够的信息来指导搜索。因此, 我们假设攻击者可以获得模型输出的软标签, 即置信度分数。

具体步骤如算法 3 所示。首先随机初始化 F 个个体, 每个个体 p 的向量维度为 $|bytes|$, 在每个世代, 计算每个个体 p_i 分类为恶意代码的置信度的负值作为适应值 fit_i, 然后进行选择、交叉和突变三种操作。最终达到世代次数, 返回分类为恶意代码置信度最低的个体 p_{best} 并生成对抗样本 x^{adv} (步骤 4)。

算法 3: SAM-GA 算法

输入: PE 恶意代码二进制文件 x, 可修改字节索引 idx, 修改字节数 γ, 显著性先验 τ, 种群规模 F, 进化代数 G

输出: 恶意代码对抗样本 x^{adv}

1. 根据显著性先验 τ 选择修改字节

$bytes \leftarrow SALIENCYSELECT(idx, \tau, \gamma)$

2. 随机初始化种群

$P \leftarrow (p_1, \ldots, p_F) \in P_{bytes}$, $P' \leftarrow \varnothing$

3. 开始种群迭代更新

FOR gen IN $[0, \ldots, G-1]$ DO

FOR p_i IN P DO

$fit_i = -f(x + p_i)$ 计算个体适应值

END FOR

$P' \leftarrow SELECT(P \cup P')$ 选择操作

$P \leftarrow CROSSOVER(P')$ 交叉操作

$P \leftarrow MUTATE(P)$ 突变操作

END FOR

$x^{adv} \leftarrow x + p_{best}$

RETURN: x^{adv}.

6　实验结果及分析

6.1　实验设置

本文的实验都是使用 python 3.9 运行在 ubuntu 20.04 的工作站上, 配置为英特尔至强 E5-2630 CPU, 32GB DDR4 内存, 以及两个 NVIDIA RTX 2080Ti 11GB GPU。

数据集及目标模型: 为了更好验证我们方法的适用性, 我们使用不同数据集上训练的 MalConv 作为目标模型。第一个是在 MLSEC2019[40]比赛中使用的预训练模型 MalConv-Ember, 其在 EMBER2018[41]数据集上进行了训练。此外, 我们从 SOREL-20M 数据集中随机选择了 10000 个样本构成了规模较小的 Mini 数据集, 训练得到 MalConv-Mini 模型。两个模型的分类精度均达到了 97%以上。

本文测试使用的 400 个恶意代码二进制文件从 Virusshare[42]上下载, 并在 Virustotal 上验证其存在恶意功能, 且测试中的恶意代码二进制文件都能被检测器正确分类。

评价指标: 为了验证恶意代码生成方法的有效性, 我们采用了规避成功率 *Evasion Rate* (*ER*)作为指标, 其代表生成的恶意代码对抗样本绕过目标检测器的能力。

$$ER = \frac{1}{n}\sum_i \mathbb{I}[f(x^{adv}) \neq f(x)] \tag{2}$$

基线方法: 本文主要研究在字节层级对恶意代

码二进制文件进行修改或添加, 以实现对目标检测器的规避。从现有采取相同操作的对抗样本生成方法中, 实验选取随机添加(Random)、良性添加(Benign)、Kruck[12]、Suciu[15]和 BFA-Enhanced[16]作为在白盒攻击的基线方法进行对比, 在黑盒设置下与随机添加 (Random)、良性添加 (Benign)、Experience-based[16]和 RAMEn[20]进行对比。

随机添加 (Random)指从均匀分布中采样字节值并附加到文件末尾。这种基线攻击衡量附加攻击过程中文件长度变化对分类的影响, 并有助于比较更复杂的附加策略对比随机添加噪音的实际收益。

良性添加 (Benign)是从良性代码的开头提取字节并将其附加。这种策略使我们能观察 MalConv 模型对在文件末尾重复使用良性字节的攻击的敏感性。其他攻击方法已在第 2 节中进行介绍。

功能性验证: 在生成对抗样本过程中, 我们通过检查 PEfile 编译库能否成功编译修改后的恶意代码, 来验证代码的结构有没有被破坏。如果生成的对抗样本编译失败, 则攻击也将被视为失败。此外, 我们从数据集中随机选择 50 个恶意代码样本, 通过 SAM 攻击创建对抗样本。将生成的对抗样本输入到 Cuckoo 沙盒中分析其行为, 并与原始样本的行为进行比较。根据我们的实验结果, 只有 4%的对抗样本与原始样本行为不同。

6.2 白盒模式 SAM 结果

在白盒设定下, 攻击者可以访问模型的内部参数, 进而利用梯度信息生成恶意代码对抗样本。本文提出基于显著性分析的白盒生成方法 SAM, 分别使用积分梯度和 Grad-CAM 两种分析手段, 得到两个结果 SAM_ig 和 SAM_cam。

如表 2 和图 7 所示, 当修改字节数较小时, SAM 方法通过优先修改显著性高的字节, 能更有效的生成能规避检测的对抗样本, 规避率成功率快速提升至 70%左右, 相较之下, 其他基线方法的规避率不超过 20%。当增加修改字节数达到 1024 字节时, SAM 方法攻击 MalConv-Ember 模型的成功率达到 72%左右, 攻击 MalConv-Mini 则超过了 76%, 均高于其他方法。分析认为, 对抗样本生成中扰动应尽量微小, 在此种限制下, SAM 方法比现有方法有明显优势。当修改字节数增加, 虽然差距缩小, 但 SAM 方法仍取得了更高的规避率。同时我们也注意到 MalConv-Mini 模型在面对对抗攻击时鲁棒性低于 MalConv-Ember, 我们猜测这是由于 Mini 数据集规模远小于 Ember 数据集, 导致模型的泛化能力不足, 进而影响鲁棒性[43]。

表 2 白盒模式下扰动量不超过 1024 字节时不同对抗方法的规避成功率

Table 2 Evasion success rate of different white-box methods when modified bytes does not exceed 1024

白盒模式	目标模型	
对抗方法	MalConv-Ember	MalConv-Mini
随机添加	0	0.4
良性添加	0	1.2
Kruck	36.93	42.27
Suciu	20.72	20.85
BFA-Enhanced	68.46	68.86
SAM	**72.97**	**76.30**

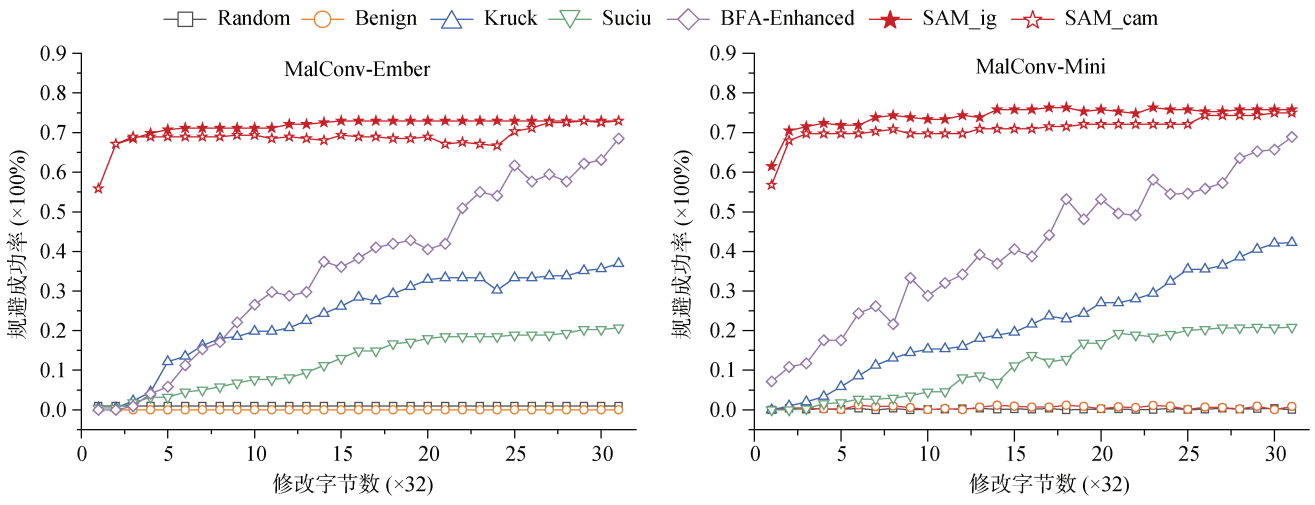


图 7 白盒模式下 SAM 与不同攻击方法的对比

Figure 7 Comparison of SAM with different attack methods in white-box mode

为了更好的体现基于显著性选择字节对攻击性能的影响, 除了基线方法外, 我们也在图 8 中比较了

从可修改序列 $idx=[d,s_1,\ldots,s_n,p]$ 中(1)单独选择部分头部数据 $partial\_dos=[d]$ (2)节空隙数据 $slack=[s_1,\ldots,s_n]$ 和(3)尾部填充数据 $padding=[p]$，以及(4)按索引顺序选择字节($index$)和(5)本文提出的 SAM 方法攻击 MalConv-Ember 的表现。观察可知，由于头部数据往往具有高显著值，修改此区域字节成功率能超过 50%。但是头部数据中可修改字节总数有限，限制了攻击性能。SAM 方法既能弥补仅修改头部可修改字节总数受限的缺点，又能在小扰动情况下随着修改字节数增加规避率快速提升，优于按索引顺序选择字节，所以基于显著性选择的策略是有效的。此外，基于积分梯度的结果要略好于基于 Grad-CAM 方法的结果，猜测认为由于 Grad-CAM 是将最后一层激活图根据卷积核大小上采样而得，导致 Grad-CAM 的分析细粒度低于积分梯度，进而性能有所降低。

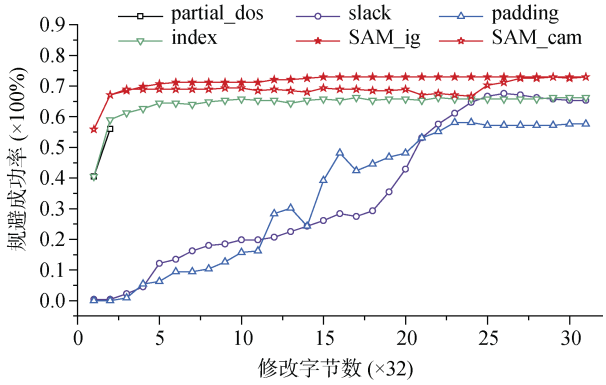


图 8　白盒模式下显著性分析的影响

Figure 8　The effect of the saliency analysis in the white-box attack

6.3　黑盒模式 SAM-GA 结果

黑盒场景是比白盒更加现实的情况，但是由于能够获取的信息更少，实现规避的难度也就更高。图 9 和表 3 中比较了本文提出的 SAM-GA 算法和不同黑盒基线方法的规避率，可以看到黑盒设定下的规避率相较白盒设定有明显降低，从 70%下降到 40%左右。但是与其他的基线方法相比，提出的 SAM-GA 仍有优势，在修改字节数不超过 1024 时，最高规避率比 Experience-based[16]高 30%、比 RAMEn[20]高 4%左右。

类似于图 8，为了更好地验证本文提出的基于显著性分析的有效性,我们也在图 10 中展示了不同字节选择策略的攻击情况。观察可知，相比修改单一区域的字节或按索引选择要修改的字节(即 RAMEn[20])，根据显著性分布先验选择修改字节取得了最高的规避率。分析认为，文件头和节空隙两个区域的可修改字节数存在上限，将限制样本的规避性能。虽然尾部填充字节的数目不存在类似限制，但由于该区域的显著性水平不高，导致攻击能力不足，往往需要填充大量字节。本文提出的基于显著性水平优先选择修改字节，有效解决了上述问题，实现了修改少量字节的情况实现对恶意代码检测器的规避。

7　结论

深度学习的恶意代码检测器缺乏可解释性且易受到对抗样本欺骗的特性，本文提出了基于显著性分析的恶意代码对抗样本生成技术，分析模型检测恶意代码的显著值分布情况，通过优先修改代码非

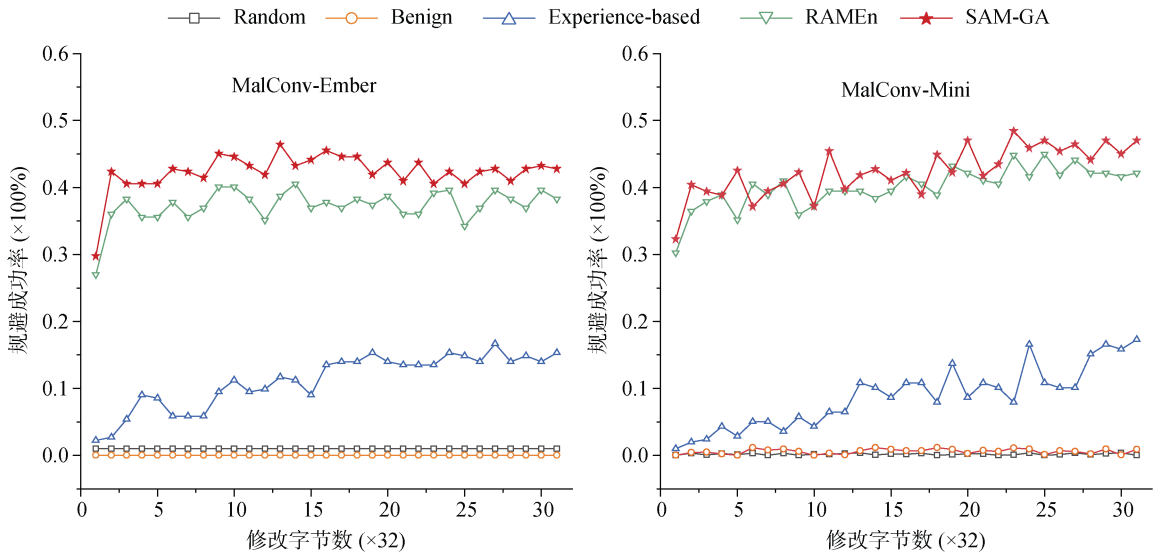


图 9　黑盒模式下 SAM-GA 与不同攻击方法的对比

Figure 9　Comparison of SAM-GA with different attack methods in black-box mode

表 3　黑盒模式下扰动量不超过 1024 字节时不同对抗方法的规避成功率

Table 3　Evasion success rate of different black-box methods when modified bytes does not exceed 1024

黑盒模式	目标模型	
对抗方法	MalConv-Ember	MalConv-Mini
随机添加	0	0.4
良性添加	0	1.2
Experience-based	16.66	17.30
RAMEn	40.52	45.03
SAM-GA	**45.49**	**48.39**

执行区域的关键字节, 实现在不改变样本原始恶意功能的条件施加对抗扰动, 并在较小扰动下有效规避恶意代码检测。实验结果表明, 在白盒模式下提出的 SAM 方法生成的恶意代码对抗样本对 MalConv 检测模型的规避成功率达到 72.9%, 黑盒模式下的规避成功率也达到了 45%, 相较其他方法有明显提升。

黑盒模式更接近现实的网络攻防应用场景且更具有挑战性, 规避成功率相较白盒模式显著降低, 我们的下一步将基于强化学习、生成对抗网络等技术, 进一步提升黑盒模式下对抗样本的规避成功率。

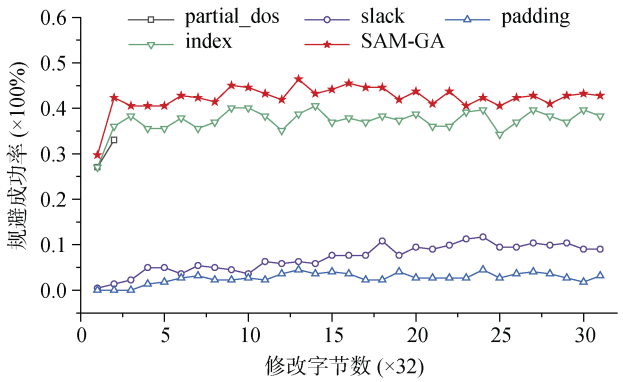


图 10　黑盒模式下显著性先验的影响

Figure 10　The effect of the saliency prior in the black-box attack

致　谢　感谢国家自然科学基金(No.62076251, No.62106281)的资助。感谢潘志松教授、各位老师和同学在实验过程中的帮助和指导, 感谢审稿专家和编辑老师的指导建议。

参考文献

[1] Kaspersky Lab. The number of new malicious files detected every day increases by 5.2% to 360,000 in 2020. https://www.kaspersky.com/about/press-releases. Oct. 2021.

[2] Sathyanarayan V S, Kohli P, Bruhadeshwar B. Signature Generation and Detection of Malware Families[M]. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008: 336-349.

[3] Banescu S, Collberg S, Pretschner A. Predicting the Resilience of Obfuscated Code Against Symbolic Execution Attacks via Machine Learning[C]. *USENIX Security Symposium*, 2017: 661-678.

[4] Calleja A, Tapiador J, Caballero J. The MalSource Dataset: Quantifying Complexity and Code Reuse in Malware Development[J]. *IEEE Transactions on Information Forensics and Security*, 2019, 14(12): 3175-3190.

[5] Idika N, Mathur A P. A survey of malware detection techniques[J]. *Purdue University*, 2007, 48(2): 32-46.

[6] Raff E, Barker J, Sylvester J, et al. Malware detection by eating a whole exe[C]. *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence,* 2018.

[7] Anderson B, McGrew D. Machine Learning for Encrypted Malware Traffic Classification: Accounting for Noisy Labels and Non-Stationarity[C]. *The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017: 1723-1732.

[8] Cui Z H, Xue F, Cai X J, et al. Detection of Malicious Code Variants Based on Deep Learning[J]. *IEEE Transactions on Industrial Informatics*, 2018, 14(7): 3187-3196.

[9] Ye Y F, Li T, Adjeroh D, et al. A Survey on Malware Detection Using Data Mining Techniques[J]. *ACM Computing Surveys*, 2018, 50(3): 1-40.

[10] Amer E, Zelinka I. A Dynamic Windows Malware Detection and Prediction Method Based on Contextual Understanding of API Call Sequence[J]. *Computers & Security*, 2020, 92: 101760.

[11] Liu X Q, Shan C, Ren J D, et al. An Intrusion Detection Method Based on Multi-Dimensional Optimization of Traffic Anomaly Analysis[J]. *Journal of Cyber Security*, 2019, 4(1): 14-26.
(刘新倩, 单纯, 任家东, 等. 基于流量异常分析多维优化的入侵检测方法[J]. *信息安全学报*, 2019, 4(1): 14-26.)

[12] Kreuk F, Barak A, Aviv-Reuven S, et al. Deceiving End-to-End Deep Learning Malware Detectors Using Adversarial Examples[EB/OL]. 2018: 1802.04528.https://arxiv.org/abs/1802.04528v3.

[13] *Wang B, Guo Y K, Qian Y G, et al. Defense of Traffic Classifiers Based on Convolutional Networks Against Adversarial Examples[J]. Journal of Cyber Security*, 2022, 7(1): 145-156.
(王滨, 郭艳凯, 钱亚冠, 等. 针对卷积神经网络流量分类器的对抗样本攻击防御[J]. *信息安全学报*, 2022, 7(1): 145-156.)

[14] Yan J, Yan J, Nie C J, et al. Method for Generating Malicious Code Adversarial Samples Based on Genetic Algorithm[J]. *Journal of Electronics & Information Technology*, 2020, 42(9): 2126-2133.
(闫佳, 闫佳, 聂楚江, 等. 基于遗传算法的恶意代码对抗样本生成方法[J]. *电子与信息学报*, 2020, 42(9): 2126-2133.)

[15] Suciu O, Coull S E, Johns J. Exploring Adversarial Examples in Malware Detection[C]. *2019 IEEE Security and Privacy Workshops*, 2019: 8-14.

[16] Chen B C, Ren Z R, Yu C, et al. Adversarial Examples for CNN-Based Malware Detectors[J]. *IEEE Access*, 2019, 7: 54360-54371.

[17] Liu X B, Zhang J L, Lin Y P, et al. ATMPA: Attacking Machine Learning-Based Malware Visualization Detection Methods *via* Adversarial Examples[C]. *The International Symposium on Qual-*

ity of Service, 2019: 1-10.

[18] Anderson H S, Kharkar A, Filar B, et al. Learning to Evade Static PE Machine Learning Malware Models *via* Reinforcement Learning[EB/OL]. 2018:1801.08917.https://arxiv.org/abs/1801.08917v2.

[19] Demetrio L, Biggio B, Lagorio G, et al. Explaining Vulnerabilities of Deep Learning to Adversarial Malware Binaries[EB/OL]. 2019: 1901.03583.https://arxiv.org/abs/1901.03583v2.

[20] Demetrio L, Coull S E, Biggio B, et al. Adversarial EXEmples[J]. *ACM Transactions on Privacy and Security*, 2021, 24(4): 1-31.

[21] Burkart N, Huber M F. A Survey on the Explainability of Supervised Machine Learning[J]. *Journal of Artificial Intelligence Research*, 2021, 70: 245-317.

[22] Zhou B L, Khosla A, Lapedriza A, et al. Learning Deep Features for Discriminative Localization[C]. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016: 2921-2929.

[23] Sundararajan M, Taly A, Yan Q. Axiomatic attribution for deep networks[C]. *International conference on machine learning, PMLR*, 2017: 3319-3328.

[24] Selvaraju R R, Cogswell M, Das A, et al. Grad-CAM: Visual Explanations from Deep Networks *via* Gradient-Based Localization[C]. *2017 IEEE International Conference on Computer Vision*, 2017: 618-626.

[25] Goodfellow I J, Shlens J, Szegedy C, et al. Explaining and Harnessing Adversarial Examples[EB/OL]. 2014: 1412.6572. https://arxiv.org/abs/1412.6572v3.

[26] Dong Y P, Liao F Z, Pang T Y, et al. Boosting Adversarial Attacks with Momentum[C]. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018: 9185-9193.

[27] Papernot N, McDaniel P, Jha S, et al. The Limitations of Deep Learning in Adversarial Settings[C]. *2016 IEEE European Symposium on Security and Privacy*, 2016: 372-387.

[28] Moosavi-Dezfooli S M, Fawzi A, Frossard P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks[C]. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016: 2574-2582.

[29] Carlini N, Wagner D. Towards Evaluating the Robustness of Neural Networks[C]. *2017 IEEE Symposium on Security and Privacy*, 2017: 39-57.

[30] Su J W, Vargas D V, Sakurai K. One Pixel Attack for Fooling Deep Neural Networks[J]. *IEEE Transactions on Evolutionary Computation*, 2019, 23(5): 828-841.

[31] Chen P Y, Zhang H, Sharma Y, et al. ZOO: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models[C]. *The 10th ACM Workshop on Artificial Intelligence and Security*, 2017: 15-26.

[32] Papernot N, McDaniel P, Goodfellow I. Transferability in Machine Learning: From Phenomena to Black-Box Attacks Using Adversarial Samples[EB/OL]. 2016: 1605.07277.https://arxiv.org/abs/1605.07277v1.

[33] Li Q Z, Guo Y W, Chen H. Practical No-Box Adversarial Attacks Against DNNS[EB/OL]. 2020: 2012.02525.https://arxiv.org/abs/2012.02525v1.

[34] Pierazzi F, Pendlebury F, Cortellazzi J, et al. Intriguing Properties of Adversarial ML Attacks in the Problem Space[C]. *2020 IEEE Symposium on Security and Privacy*, 2020: 1332-1349.

[35] Microsoft, Inc. 2020. PE Format. https://docs.microsoft.com/en-us/windows/win32/debug/pe-format. Oct. 2020.

[36] Tim Paterson. 1983. An inside look at MS-DOS[M], 1983: 230.

[37] Sayres R, Taly A, Rahimy E, et al. Using a Deep Learning Algorithm and Integrated Gradients Explanation to Assist Grading for Diabetic Retinopathy[J]. *Ophthalmology*, 2019, 126(4): 552-564.

[38] Sanyal S, Ren X. Discretized Integrated Gradients for Explaining Language Models[EB/OL]. 2021: 2108.13654.https://arxiv.org/abs/2108.13654v1.

[39] Kapishnikov A, Venugopalan S, Avci B, et al. Guided Integrated Gradients: An Adaptive Path Method for Removing Noise[C]. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021: 5048-5056.

[40] MLSEC2019. https://github.com/endgameinc/malware evasion competition.

[41] Anderson H S, Roth P. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models[EB/OL]. 2018: 1804.04637.https://arxiv.org/abs/1804.04637v2.

[42] Virusshare. https://virusshare.com.

[43] Bubeck S, Sellke M. A universal law of robustness via isoperimetry[J]. *Advances in Neural Information Processing Systems*, 2021, 34: 28811-28822.

附录 A MalConv 恶意代码检测模型

MalConv 模型包含一个 8 维嵌入层和一个一维门控卷积。嵌入层作为一个特征映射函数, 将每个字节映射为嵌入空间中的固定长度的特征向量, 使用嵌入空间的特征向量代替原始字节值学习更高层次的表征。然后将嵌入向量输入到 2 个平行的门控卷积层中, 将输出进行相乘后传递给非线性的 RELU 激活单元。将每个字节作为输入序列中的一个单元来处理的话, 远远超过之前所有基于神经网络的序列分类器的输入的长度, 因此该方法使用步长和卷积核大小为 500 的门控卷积层对特征进行降维, 减小计算成本并获得更广的视野域。同时, 对全连接层最大池化处理后, 使得系统能够评估整个文件中这些指标的相对强度, 从而找出重要的全局组合。

训练过程中, 嵌入层与门控卷积一起优化, 使得语义上表现出相似行为的字节在特征空间中的距离更近。通过上述的网络模型设计, MalConv 实现了可控的计算成本, 在寻找性能更好的体系结构的时候, 在增加模型中训练参数的数量的同时, 控制了内存占用量和满足实际要求的训练时间。加入门控卷积层和最大池化层的设计使得模型可以考虑局部和全局上下文信息以及具备更好的可解释性。

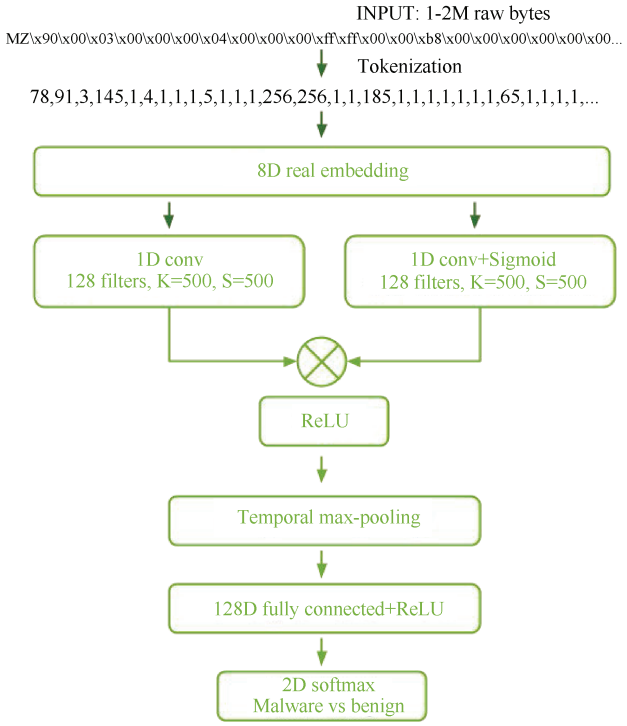


图 A1　MalConv 网络结构

Figure A1　MalConv's network structure

附录 B 积分梯度和 Grad-CAM

在积分梯度算法中, 模型需同时满足敏感度(Sensitivity)和实现不变性(Implementation Invariance)两个公理:

1、敏感度 (Sensitivity): 当输入非基线值时获得有状态的预测, 当选取两个不同的输入值(相对于基线值而言)时, 模型输出不同的预测。

2、实现不变性 (Implementation Invariance): 对于两个函数等效的网络(输入相同则输出相同), 归因应当一致。

积分梯度算法结合了直接梯度和基于反向传播的归因技术的设计思想, 将输入的第 i 个特征的归因定义为: 从基线 x_i' 到输入 x_i 之间的直线路径的路径积分, 通过在基线值和输入值之间选取了无限多个积分点进行积分加和, 显著值 s_i 可表达为:

$$s_i = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x_i' + \alpha \times (x_i - x_i'))}{\partial x_i'} \mathrm{d}\alpha \tag{3}$$

从上式的表达可以看出集成梯度算法仅仅考虑了模型的输入输出且函数处处可微, 不需要模型内部细节的参与, 因此集成梯度算法是满足实现不变性的。

类激活映射 (Class Activation Mapping, CAM)技术旨在一个提供可视化卷积神经网络的工具, 使用类激活映射可以清楚的观察到网络关注输入样本的区域。在类激活映射方法中, 模型的全连接层被替换为全局平均池化层, 某类 C 的置信度 S 可以表示为其全局平均池化层输出的特征图 L 的线性组合, 显著图中每个位置 (i, j) 的显著值 V 可以表示为:

$$\begin{aligned} S^c &= \sum_k w_k^c \cdot \sum_i \sum_j L_{ij}^k \\ V_{ij}^c &= \sum_k w_k^c L_{ij}^k \end{aligned} \tag{4}$$

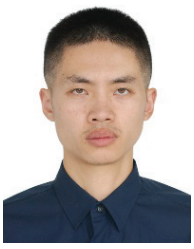
但该方法需要改变网络结构, 把全连接层改成全局平均池化层, 不利于训练。因此提出了 CAM 的泛化形式 Grad-CAM, 用梯度的全局平均来计算权重, 计算公式如下:

$$w_k^c = \sum_i \sum_j \alpha_{ij}^{kc} \cdot RELU\left(\frac{\partial S^c}{\partial L_{ij}^k}\right) \tag{5}$$

求得类别所有特征图的权重后, 求其加权却和就可以得到热力图。

詹达之　于 2017 年在国防科技大学系统工程专业获得硕士学位。2024 年在陆军工程大学获得博士学位。研究领域为恶意代码、对抗样本。Email: zhangaga93@outlook.com

孙毅　于 2018 年在陆军工程大学获得硕士学位, 2022 年在陆军工程大学获得博士学位。现为国防科技大学助理研究员, 研究方向包括自然语言处理, 领域适应。Email: sunyi_lgdx@sina.com



张磊　于 2018 年在陆军工程大学获得硕士学位, 2022 年在陆军工程大学获得博士学位。现为军事科学院的助理研究员。研究方向包括数据工程, 人工智能。Email: zhanglei@aeu.edu.cn

刘鑫　于 2018 年在陆军工程大学获得计算机科学硕士学位, 2022 年在陆军工程大学获得博士学位。研究方向为优化算法。Email: liuxin@aeu.edu.cn

郭世泽 于 1994 年在哈尔滨工业大学电子工程专业获得博士学位。现为陆军工程大学教授。研究领域为信息技术和信息安全。Email: shize_aeu@outlook.com

潘志松 于 2003 年在南京航空航天大学大学计算机科学与技术专业获得博士学位。现为陆军工程大学教授。研究领域为深度学习、机器学习和模式识别。Email: panzhisong@aeu.edu.cn