**摘要**

现代商业防病毒系统日益依赖机器学习技术以应对新恶意软件的激增。然而众所周知，机器学习模型易受对抗样本（AEs）攻击。先前研究表明，机器学习恶意软件分类器对白盒对抗攻击具有脆弱性。但由于商业杀毒软件中的机器学习模型通常不对外公开，且仅返回硬分类标签，因此以纯黑盒方式评估机器学习模型和实际杀毒软件的健壮性更具实践意义。我们提出了一种基于黑盒强化学习（RL）的框架，用于生成针对PE格式恶意软件分类器和杀毒引擎的对抗样本。该框架将对抗攻击问题建模为多臂老虎机问题，在利用成功攻击模式与探索攻击多样性之间寻求最优平衡。相比现有框架，我们的改进体现在三方面：1）通过将生成过程建模为无状态过程，限制探索空间以避免组合爆炸；2）鉴于有效载荷在对抗样本生成中的关键作用，设计了对成功载荷的复用机制；3）最小化对抗样本的修改量以确保强化学习中的奖励正确分配，同时有助于溯源规避机理。实验表明，本框架的黑盒规避率显著高于现有方案：对两种前沿机器学习检测器的规避率达74%-97%，对商业杀毒软件在纯黑盒环境下规避率达32%-48%。研究还发现，基于机器学习的分类器间的攻击可迁移性，显著高于纯机器学习模型与商业杀毒软件之间的迁移性。

**关键词**对抗攻击，强化学习，神经网络，恶意软件分类

1 引言

恶意软件攻击始终是当今用户面临的最紧迫安全问题之一。最新研究表明，2019年前三季度已报告至少72亿次恶意软件攻击和1.519亿次勒索软件攻击。随着COVID-19疫情暴发，攻击频率再创新高。传统的基于特征检测方法已无法应对新型恶意软件的激增，商业杀毒公司因此开始采用机器学习技术[7,66]。基于机器学习的检测系统具有可扩展性和高效性，能有效应对海量恶意软件威胁。自2001年首篇关于机器学习检测恶意软件的论文[61]问世以来，学术界涌现出大量利用机器学习预测恶意内容的研究成果，其中多数标榜高准确率并声称能检测训练阶段未见过的新型恶意软件[5,15,55,56,60]。

然而，研究也表明基于机器学习的检测系统存在严重安全漏洞：攻击者仅需对恶意软件进行细微修改即可轻易绕过检测[3,4,10,13,14,17,19-22,24,26,27,30,33,35-37,39,40,42,44,45,49,52,53,57,58,63-65,68,69]。即便是Cylance等商业杀毒系统，也被证实可能因简单的对抗攻击而失效[6]。

自2014年以来，对抗攻击与防御相关研究已超过1400篇。然而，这些研究主要集中于特征值领域。针对恶意软件样本的对抗攻击与特征值领域的攻击存在本质差异：在图片领域，攻击者可以修改任意像素值，只要变化幅度受限于𝐿𝑝范数；但对于恶意软件样本，即便仅修改一个字节，也可能破坏有效PE文件格式或影响原有恶意功能。正因如此，攻击者通常不会直接修改PE文件的原始字节，而是构建一组操作指令——每个指令都能在不破坏原有功能的前提下对恶意软件样本进行变形。例如，典型操作包括添加新的冗余节区（即在节区表中新增条目，并在文件末尾附加节区内容）。因此，对抗样本生成问题就转化为寻找能够导致错误分类的正确操作及相应内容。

针对静态恶意软件分类器的对抗攻击并非新兴课题。研究者已提出多种技术生成对抗样板，包括遗传编程[18,68]、蒙特卡洛树搜索[54]以及Deep Q Learning[4]。尽管部分研究[54,68]分别针对PDF恶意软件和源代码溯源问题，但其通用算法可扩展至PE恶意软件检测领域。

现有技术虽被证实有效，但我们发现存在若干局限性。首先，现有方法采用有状态建模生成对抗样本，即各操作之间存在依赖关系。这种建模方式虽具普适性，但由于搜索空间庞大，通常难以有效训练。我们观察到PE恶意软件变形操作大多具有独立性，因此采用无状态建模方法，显著降低学习难度并提升对抗样本生成效率。其次，现有技术主要学习决策策略以确定下一步操作，所需内容则随机生成。例如添加新节区时，仅填充随机内容。然而内容与操作同等重要——若某操作-内容组合在某个对抗样本中有效，该组合很可能具有普适性。因此应将操作及其内容作为整体建模。第三，当成功生成对抗样本时，现有技术会为所有参与操作分配奖励。经评估发现，实际起关键作用的操作通常仅1-2个，其余皆为冗余。为冗余操作分配奖励将干扰学习过程。

基于上述洞见，我们提出名为MAB-Malware的开源强化学习框架，用于生成PE恶意软件的对抗样本。通过将每个操作-内容组合视为独立的老虎机，我们将该问题建模为经典多臂老虎机（MAB）问题：将每个老虎机的奖励建模为贝塔分布，并采用汤普森采样策略选择下一步操作与内容，在利用已知有效模式与探索新策略之间实现动态平衡。

我们设计了操作最小化处理机制——通过剔除冗余操作，并将关键操作进一步分解为更细粒度的微操作，从而精简对抗样本。随后仅对这些核心微操作分配奖励。该机制同时有助于解析样本规避检测的根本原因。

本文的主要贡献可归纳为以下三点：

• 我们系统分析了现有黑盒对抗样本生成算法，针对有状态与无状态建模、内容相关与内容无关建模、冗余操作与核心操作等关键维度提出创新性洞见。  
• 论证了无状态建模与内容感知建模在生成PE恶意软件对抗样本中的优越性，并揭示操作最小化处理的关键作用。  
• 基于上述设计原则，提出并实现首个基于多臂老虎机（MAB）的强化学习框架，专门用于生成PE格式对抗样本。  
• 通过5000个PE恶意软件样本对两个主流机器学习模型及三款商业杀毒引擎进行大规模评估：MAB-Malware对机器学习模型的规避率超75%，显著优于现有黑盒对抗生成算法；在商业杀毒引擎中也展现明显改进。  
• 基于操作最小化分析，深入探究检测规避的根本成因。实验表明商业杀毒引擎的静态分类器易受样本细微改动影响。同时揭示对抗攻击在基于机器学习的分类器间具有高可迁移性（超80%），而纯机器学习模型与商业杀毒引擎间的迁移率极低（不足7%）。

为促进该领域的后续研究，我们承诺将根据学术研究需求，在通过申请审核后向研究人员开源共享本研究的框架代码及对抗性恶意软件样本数据集。

**2 问题定义**

**2.1 威胁模型**

我们遵循Carlini等人[8]的研究方法，从对抗目标、对抗能力与对抗知识三个维度构建威胁模型。

**对抗目标**

攻击者旨在通过对恶意软件样本进行可控修改，规避静态PE恶意软件分类器的检测。本研究聚焦Windows PE格式恶意软件，PDF或Android等其他类型恶意软件不在研究范围内。该攻击属于非定向攻击（untargeted attack）：由于分类任务仅为二分类（良性/恶意），不涉及特定恶意软件家族识别，攻击目标仅需使恶意样本被误判为良性。

**对抗能力**

本研究假设攻击者无法干预分类器的训练过程（如向训练集注入污染数据）。同时，攻击者不可对输入数据进行任意修改。与图像对抗攻击中要求"细微修改以保持视觉不可察觉性"不同，恶意软件对抗攻击的约束在于必须保留原始恶意功能。若修改虽能规避检测但破坏恶意功能，则视为攻击失败。

**对抗知识**

根据攻击者对模型的认知程度，攻击可分为两类：1) 白盒攻击——完全掌握模型内部信息；2) 黑盒攻击——不了解模型细节，仅通过有限次数的查询获取分类结果（概率分值或硬分类标签）。

本研究假设攻击者仅具备黑盒访问权限：不了解分类器内部机制，可进行有限次数的检测查询，且当样本被判定为恶意时能够获取分类器响应。

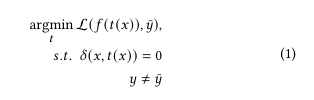
2.2 问题定义  
本研究聚焦于三种前沿机器学习分类器及三款头部商业杀毒软件的静态分类器。

我们的核心目标是自动生成恶意软件分类器的对抗样本，并解析检测规避的根本成因。该问题可分解为两个子问题：对抗样本生成与特征可解释性分析。

具体而言，我们需对恶意软件样本实施可控修改，使其被分类器误判为良性软件，同时确保原始恶意功能完整。在图像领域的白盒攻击中，原始图像的修改受限于𝐿2和𝐿∞范数（衡量像素变化的数学指标），以保证人类视觉不可察觉性。然而在恶意软件分类领域，只要恶意行为保持不变，普通用户通常无法察觉差异。这正是先前针对恶意软件的黑盒攻击[4,11,23]在生成对抗样本时未要求最小化修改量的原因。

但我们发现最小化修改仍具关键价值，主要基于三点考量：1) 揭示可复用的核心操作及其有效载荷——这些要素可移植至其他样本以提高攻击成功率；2) 定位导致规避的关键特征变化，确保分类器不依赖表面特征；3) 降低生成损坏二进制文件的风险。在黑盒设定下，我们通过最小化生成对抗样本所需的操作序列实现这一目标，具体包括：剔除冗余操作、替换可能大幅改变检测特征的修改操作。

设X为恶意软件数据集，𝑓为恶意软件分类器，将样本𝑥 ∈ X映射至分类标签𝑦 ∈ {0,1}（0代表良性，1代表恶意）。我们构建操作集合A = {𝑎₁,𝑎₂,...𝑎ₙ}，用于对恶意软件样本实施扰动。对抗样本生成的目标函数定义如公式(2)所示：通过应用转换函数𝑡（即从集合A中采样的操作序列），生成对抗样本𝑥′ = 𝑡(𝑥)。损失函数L(𝑓(𝑡(𝑥)), ȳ)衡量分类器𝑓对转换样本𝑡(𝑥)的预测结果与良性标签ȳ之间的偏差。转换函数𝑡需满足约束条件：𝑡(𝑥)不改变𝑥的原始功能，即转换前后的功能差异𝛿(𝑥,𝑡(𝑥))等于0。



**3 研究动机**

本节首先探讨现有对抗样本（AE）生成技术及其局限性，进而阐释我们提出基于多臂老虎机（MAB）方法的理论依据。

3.1 现有对抗样本生成技术  
**深度Q学习（DQN）**  
Anderson等人[4]提出应用深度强化学习（RL）生成PE格式恶意软件的对抗样本以绕过机器学习检测。该方法首先定义操作集合（文件变异操作），包括修改PE头、追加覆盖字节、加壳与脱壳等。智能体基于策略与环境状态选择下一步操作。当成功生成规避样本时，所有参与操作（包括早期未立即获得奖励的操作）在给定状态下均获得激励。

**遗传编程**  
Demetrio等人[18]提出基于遗传编程的黑盒攻击方法生成PE恶意软件对抗样本。该方法将问题形式化为约束最小化问题，在规避概率与注入载荷规模间寻求平衡：首先从良性二进制文件中提取内容片段，随机选取部分追加至目标恶意样本尾部；定义适应度函数为置信度分数与注入载荷规模之和；每轮迭代选择适应度最低的变体，通过选择、交叉（cross-over）、变异（mutation）三阶段产生新种群。交叉与变异确保新种群与先前变体充分差异化。

Xu等人[68]同样采用遗传编程生成PDF恶意软件对抗样本：定义包含删除、插入、替换PDF树结构对象等变异操作集合；以置信度分数作为适应度标准，执行选择与变异（不含交叉）。与Demetrio方法不同，该研究保留并复用成功攻击轨迹（包含操作序列及每个操作使用的内容），用于新样本生成。

**蒙特卡洛树搜索**  
Quiring等人[54]提出一种基于蒙特卡洛树搜索（MCTS）的方法，用于干扰源代码作者归属分类。尽管该研究不涉及恶意软件分类领域，但其核心思想具备普适性。他们定义了一组代码风格转换操作，并构建蒙特卡洛搜索树——每个节点代表代码的一个变体，每条边代表一个操作。每轮迭代中，选择平均分类器得分最高的节点，从该节点生成一系列转换序列，并将观测到的分类器得分从叶节点反向传播至根节点。此过程持续迭代直至生成规避样本，从而将对抗样本生成转化为路径搜索问题，目标是在树中找到导致错误分类的路径。

**3.2 理论洞见**  
尽管现有技术已展现一定有效性，我们通过观察提出若干关键见解，为开发更优的对抗样本生成技术奠定理论基础。

**有状态建模 vs. 无状态建模**  
现有技术采用有状态建模方式生成对抗样本：旨在每轮迭代中寻找最优状态，根据策略选择当前状态下的最优操作，并通过转换样本进入下一状态。本质上，此类方法试图找到导致规避的最优状态路径。这种建模方式适用于复杂任务（如AlphaGo围棋博弈与电子游戏通关），因其成功路径往往具有深度依赖关系，但也意味着训练过程通常困难且耗时。

我们的核心洞见在于：针对PE恶意软件的对抗样本生成无需采用有状态建模。由于对PE文件实施功能性无损修改的难度较高，现有操作多为粗粒度且相互独立。例如，移除调试信息、节区重命名、新增节区等操作可任意顺序执行，最终生成的二进制文件结果相同。即便存在操作间依赖（如为PE文件添加多个节区时顺序影响结果），我们论证此类依赖关系强度有限。根据第5.4节对机器学习模型与杀毒引擎的评估结果，在剔除冗余操作后，通常仅需1-2个关键操作即可生成对抗样本。这表明操作间依赖关系（若存在）至少在当前评估对象中表现微弱。

基于此洞见，我们确信对PE恶意软件对抗样本生成采用无状态建模具有合理性。相较于有状态建模，无状态建模将每个操作视为独立单元，可实现更快速的学习过程与更高效的对抗样本生成。为此，我们提出采用经典强化学习模型——多臂老虎机（MAB）[41]，将每个操作建模为独立的老虎机。该模型通过有限次试验估计各操作的规避概率，并基于概率估计选择最优操作以最大化整体收益（即在有限尝试次数内生成尽可能多的对抗样本）。

**内容建模**  
许多PE文件操作需关联特定内容。例如，添加新节区需指定填充内容，重命名节区需提供新名称。我们的第二项洞见在于：内容与操作同等重要。若某操作-内容组合在某个对抗样本中有效，该组合很可能具备持续有效性。

遗憾的是，现有技术（除Xu等人[68]外）均未充分考虑内容因素。这些方法仅学习决策策略以确定下一步操作，所需内容则随机生成。例如，若策略指定下一步操作为"新增节区"，则随机填充内容（从数据池选取）。Xu等人[68]虽考虑内容因素，但未充分挖掘其价值（仅首次生成时复用轨迹）。

本研究的基于MAB框架将操作-内容组合作为整体单元（即一台老虎机）建模：若新发现的组合能有效生成对抗样本，则将其作为新老虎机存入机器池。当其他样本再次选择该机器时，可复用相同内容。

**精准奖励分配**  
奖励分配机制是上述所有对抗样本生成技术的核心要素。当成功生成对抗样本时，正向奖励将被分配至对应状态或操作序列。但如前所述，并非所有操作都对生成该样本起关键作用。根据第5.4节的评估结果，大多数情况下仅1-2个操作为核心要素，其余皆属冗余。因此，为所有参与生成对抗样本的操作分配奖励会导致强化学习模型精度下降。

我们的第三项洞见在于：应通过剔除冗余操作，仅对核心操作分配奖励以实现精准奖励。此举可显著提升模型学习效率与准确性。

为识别对抗样本中的关键操作，我们设计最小化处理流程（如图1所示）。假设原始恶意样本𝑥位于特征空间的恶意区域。通过连续施加操作𝑎₁、𝑎₂、𝑎₃生成样本𝑥₁₂₃，其成功进入良性区域成为对抗样本。在最小化阶段：

1. **剔除冗余操作**：移除操作𝑎₁后，生成样本𝑥₂₃仍能规避检测，故𝑎₁为冗余；移除操作𝑎₂后，生成样本𝑥₁₃无法规避，故𝑎₂为核心操作。
2. **操作细粒度分解**：将关键操作分解为微操作。例如，𝑎₂可替换为微操作𝑎₂′，𝑎₃可分解为𝑎₃′或𝑎₃′′。生成样本𝑥₂′₃、𝑥₂′₃′、𝑥₂′₃′′进行验证。
3. **生成精简对抗样本**：最终获得样本𝑥₂′₃′′，其仅需微操作序列（𝑎₂′,𝑎₃′′）即可实现规避。此时仅需对核心微操作𝑎₂′与𝑎₃′′分配正向奖励

**4 MAB-MALWARE**

**4.1 框架结构**

如图2所示，MAB-Malware框架工作流程包含两大核心模块：二进制重写器（Binary Rewriter）与动作最小化模块（Action Minimizer）。

二进制重写器通过汤普森采样（Thompson sampling）从操作集合A中选取操作序列𝑡，将原始恶意样本𝑥重写为变体𝑥′ = 𝑡(𝑥)。若𝑥′能绕过目标分类器检测，动作最小化模块则继续从𝑡中剔除冗余操作，生成最小化规避样本𝑥′\_{min}，并推断规避成因。

功能验证阶段：框架验证𝑥′{min}的恶意功能完整性。若功能受损，原始样本𝑥将重新加入处理队列；若功能完整，则输出𝑥′{min}作为最终对抗样本。

**4.2 二进制重写器  
4.2.1 操作集与特征分类**  
如表1所示，我们定义用于生成恶意软件对抗样本的操作集合。每个操作均针对分类器可能用于检测的特定特征实施修改。通过分析EMBER等开源恶意软件检测器，推测商业杀毒软件可能采用类似特征。这些特征分为三类：

1. **基于哈希的签名**
   * 文件哈希
   * 节区哈希
2. **基于规则的签名**
   * 节区数量
   * 节区名称
   * 节区填充
   * 调试信息
   * 校验和
   * 数字证书
   * 代码序列
3. **基于数据分布的特征**
   * 字节直方图
   * 字节熵直方图

**二进制重写器**  
**宏操作实现**  
我们基于pefile库实现了Anderson等人[4]提出的大部分操作，并修复了可能破坏功能的多种边界情况。同时采用Pappas等人[48]提出的代码随机化操作（CR）——该技术最初作为防御面向返回编程（ROP）攻击的方法被提出。

**微操作定义**  
若某宏操作𝑎修改恶意软件样本的特征集𝐹 = {𝑓₁, 𝑓₂,...𝑓ₖ}，则仅改变𝐹子集的操作即为𝑎的微操作。我们为OA（覆盖追加）、SP（节区填充）、SA（新增节区）、SR（节区重命名）、CR（代码随机化）等宏操作共实现5类微操作。如表2所示，本框架所有操作及其对应影响特征如下：

以SP操作为例（在节区尾部追加内容）：通过表2可知，SP操作影响文件哈希（𝐹₁）、节区哈希（𝐹₂）及基于规则的签名（节区填充数据𝐹₅）。其中𝐹₁、𝐹₂随文件或节区内容修改而变化，𝐹₅仅在填充内容包含基于主体数据的签名时受影响。因此，SP的微操作SP1（在节区尾部追加1字节）不影响𝐹₅（见表2）。同理，OA1操作（在二进制文件尾部追加1字节）作为SP的微操作，仅影响𝐹₁。

**4.2.2 多臂老虎机（MAB）建模**  
在对抗样本生成问题中，每个携带具体内容的操作对生成规避样本的有效性具有不确定性。我们将每个操作-内容组合视为独立的老虎机𝑚，每台机器根据其特定概率分布提供随机奖励。目标是通过有限次尝试，最大化一系列操作-内容组合所获奖励总和。由于尝试次数受限，每次选择需在"利用"（选择预期奖励最高的组合）与"探索"（获取其他组合的奖励信息）间进行权衡。

多臂老虎机问题[41]作为经典强化学习问题，通过将有限资源分配至多个竞争选项（每台老虎机对应一个选项）实现收益最大化。在此过程中，各选项的属性通过资源分配过程逐步学习。每个操作-内容组合即被建模为一台独立的老虎机，其奖励机制反映该组合对生成对抗样本的有效性。

**4.2.3 汤普森抽样**  
在本任务中，我们面临**延迟反馈问题**。当评估商业杀毒软件的静态检测模块时，需将生成的样本复制至装有杀毒软件的虚拟机并等待扫描结果。该过程耗时数秒至数分钟不等，若采用确定性算法（如UCB1或贝叶斯UCB），在结果返回前始终选择当前估值最高的选项，将因信息滞后导致低效尝试。为此，我们采用**汤普森抽样算法**[67]——相较于确定性算法，该算法在延迟反馈环境下具备更强鲁棒性[12]。

**概率建模**  
假设每台老虎机𝑚的奖励服从其特有的贝塔分布[1]。贝塔分布是由两个正参数𝛼、𝛽定义的连续概率分布，即𝑚 ∼ Beta(𝛼,𝛽)。形式化而言，每台机器𝑚的奖励服从伯努利分布𝑟(𝑎) ∼ 𝐵𝑒𝑟𝑛𝑜𝑢𝑙𝑙𝑖(𝜃)，其中期望奖励𝜃为固定未知值。拉动机器𝑚时，以概率𝜃获得奖励1，以概率1−𝜃获得0。

**参数更新机制**  
初始化时，每台机器的𝛼=1，𝛽=1。当生成规避样本时：

* 若某操作无法被最小化器进一步优化（最小化流程详见下节），其𝛼值加1；
* 否则，𝛽值加1。  
  𝛼与𝛽分别对应操作的成功与失败计数。

**动态平衡策略**  
在每次动作选择迭代中，从每台机器的𝐵𝑒𝑡𝑎(𝜃;𝛼,𝛽)分布中采样一个值，选择采样值最高的机器作为下一步操作。当机器的𝛼、𝛽值较小时，其不确定性较高，即使平均奖励低于其他机器，仍可能因采样值较大被选中（促进探索）；随着试验次数增加，𝛼、𝛽值增大，不确定性降低，高奖励机器被优先选择（促进利用）。汤普森抽样通过概率机制自动实现探索与利用的最优平衡。

**新机器生成机制**  
当生成并最小化规避样本后，除更新现有机器参数外，还需将新机器加入机器池：若核心机器包含有效内容，则创建新机器（初始𝛼=1，𝛽=1）。新机器因高不确定性易被选中，且复用成功内容提升攻击效率。

**4.2.4 二进制重写器工作流程**  
如算法1所示，二进制重写器的工作流程如下：针对种子恶意样本𝑠，目标生成最小化规避样本𝑥′*{min}，使其成功规避目标分类器检测（𝑓(𝑥′*{min})=0）且仅修改最少量特征。

**初始化阶段**

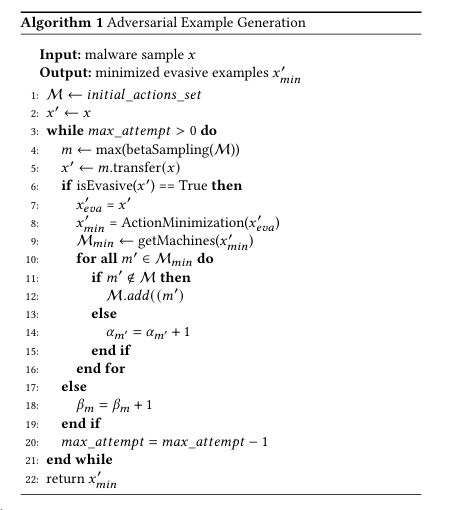
1. 将原始机器加入机器列表M，每台机器的𝛼值与𝛽值初始化为1；
2. 当机器需要内容时，从随机内容池中选取。

**迭代生成阶段**  
每轮迭代中：

* 基于每台机器的贝塔分布采样数值；
* 选择采样值最高的机器，施加对应操作至当前样本；
* 持续迭代应用操作直至生成规避样本或达到最大尝试次数𝑚𝑎𝑥\_𝑎𝑡𝑡𝑒𝑚𝑝𝑡。

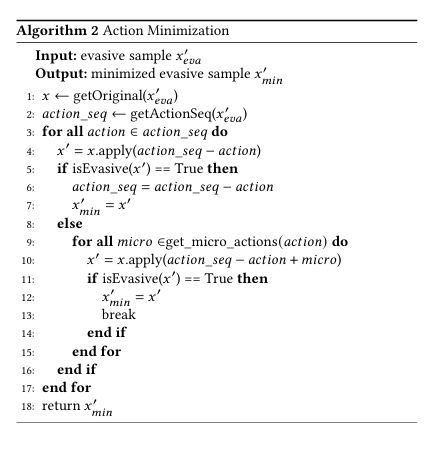
**结果处理机制**

* **成功生成规避样本**：
  1. 调用动作最小化器移除冗余机器；
  2. 若剩余机器包含新内容，则创建新机器𝑚′（初始𝛼=1，𝛽=1）；
  3. 若为现有机器，则提升该机器及其父机器（相同操作但随机内容）的𝛼值。
* **生成失败**：  
  提升当前机器的𝛽值以降低后续选择概率。



**4.3 动作最小化模块**  
动作最小化模块通过以下两步生成"最小化"规避样本：

1. **剔除冗余操作**：移除对抗样本生成过程中非必要的操作；
2. **操作细粒度替换**：以微操作替代宏操作，确保仅修改最少量检测特征即可实现规避。



如**算法2**所示，针对规避样本的操作序列𝑞 = (𝑎₁,𝑎₂,...)，动作最小化流程如下：

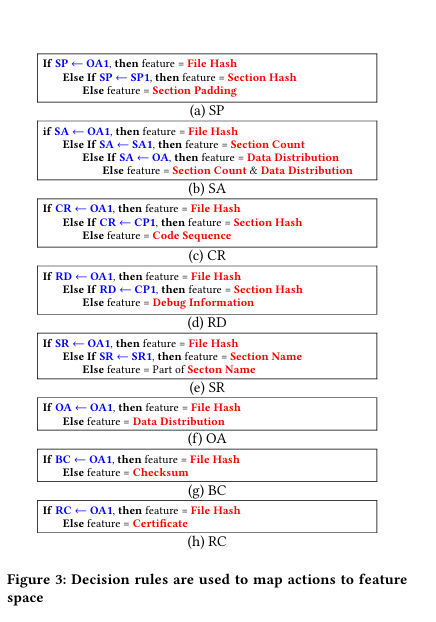
**冗余操作剔除**  
依次移除每个操作𝑎𝑖，生成新样本𝑥′。若𝑥′仍可规避检测，则判定𝑎𝑖冗余并剔除；否则保留。

**宏操作细粒度替换**  
以新增节区（SA）操作为例：

* **宏操作影响特征**：文件哈希（𝐹₁）、节区哈希（𝐹₂）、节区填充签名（𝐹₅）；
* **微操作替换策略**：
  1. 尝试用覆盖追加1字节（OA1）替换SA（仅影响𝐹₁）；
  2. 若仍可规避，则确认规避由破坏文件哈希（𝐹₁）导致；
  3. 若失败，改用节区尾部追加1字节（SP1）替换SA（影响𝐹₁、𝐹₂）；
  4. 成功则规避源自节区哈希（𝐹₂）失效，否则归因于节区填充签名（𝐹₅）被破坏。

**防御视角的根因分析**  
通过宏操作分解为微操作的过程，可定位分类器最脆弱的特征维度。如图3所示，该方法能精确识别：

* 文件哈希校验机制的失效；
* 节区哈希检测的漏洞；
* 节区填充签名规则的缺陷。



**5 实验评估  
5.1 实验设置**  
**数据集**  
本研究针对Windows PE格式二进制文件生成对抗样本。为确保生成样本的可执行性与功能完整性，PE文件格式及其约束条件必须保持合规。

**恶意软件筛选标准**  
从VirusTotal平台随机选取5000个样本，满足：

1. 超80%杀毒引擎判定为恶意；
2. 在Cuckoo沙箱中运行呈现恶意行为。

**数据清洗**  
排除Visual Basic（VB）程序，原因如下：

* 代码随机化实现[48]依赖的IDA Pro无法为VB程序生成控制流图（CFG）；
* 此类实现问题留待未来研究解决。

**实验环境配置**

实验在Microsoft Azure云平台的20台虚拟机上执行，每台配置为Standard D2s v3（2个虚拟CPU，8 GiB内存）。所有脚本（包括二进制重写器、动作最小化模块及结果分析工具）均基于Python 3.6.9实现。二进制重写器依赖pefile库与IDA Pro 6.8。测试所用商业杀毒软件均采用免费版本与默认配置。

安全隔离措施

所有网络流量均路由至宿主机的InetSim实例（模拟网络服务），以防止恶意软件感染其他设备，并确保实验稳定性和可复现性。

测试对象选择：

**商业杀毒软件**：选取三款头部商业杀毒产品（匿名化为AV1、AV2、AV3），分别部署于Windows 7（版本6.1 Build 7601，SP1）的Azure虚拟机上；

**EMBER[5]**：作为一个开源机器学习分类器，采用基于树形分类器模型LightGBM，通过2350维特征向量检测恶意软件。特征包括原始特征（字节直方图、字节熵直方图、字符串等）与解析特征（通用文件信息、头部信息、节区信息、导入/导出表等）。实验代码基于2019年机器学习安全对抗竞赛（MLSEC）实现[46]；

**MalConv[55]**：直接从恶意软件原始字节中学习的神经网络检测模型，采用MLSEC 2019版实现[46]。

**5.2 对抗样本生成**  
**与现有框架的对比实验**  
我们将MAB-Malware与两款主流对抗攻击框架进行对比：

1. **SecML-Malware**
   * SecML Python库的恶意软件检测对抗插件，集成多种攻击方法（含黑盒与白盒攻击）；
   * 本实验采用其基于遗传编程的黑盒攻击（GAMMA）；
   * 支持基于置信度分数与硬标签（仅返回良性/恶意标签）两种选择策略，本实验根据威胁模型假设（攻击者无法获取置信度分数）仅评估基于硬标签的攻击模式。
2. **Gym-Malware**
   * 基于OpenAI gym构建的强化学习恶意软件对抗环境；
   * 智能体通过操作PE文件获取奖励信号学习规避杀毒软件检测；
   * 为评估强化学习效果，支持强化学习智能体动作选择与随机动作选择两种模式。

针对MalConv与EMBER两个机器学习模型，我们定义规避率（Evasion Rate）为：  
𝑅ₑ = 𝑁ₑ/𝑁₅  
其中：

* 𝑁ₑ：成功规避样本总数；
* 𝑁₅：目标模型可检测的原始样本总数。

实验参数设置  
为公平对比，三款框架采用：

1. 相同数据集：来自VirusTotal的5000个样本；
2. 相同预训练模型：基于2019年机器学习静态规避竞赛（MLSEC）[46]提供的MalConv与EMBER模型。

每个实验重复运行五次以计算平均值，确保结果统计显著性。

从图4可见，本研究的MAB-Malware框架效果显著优于其他方法：对MalConv模型的规避率达97.72%，对EMBER模型达74.4%。相较之下，SecML-Malware（GAMMA硬标签模式）的规避率分别为63.6%与50.0%。值得注意的是，即便在获取置信度分数的理想条件下（虚线），SecML-Malware的规避率提升有限甚至出现下降。

Gym-Malware表现最差，其强化学习模式（实线）与随机操作模式（虚线）的规避率几乎持平。这表明其深度Q学习模型未能有效学习规避策略。根本原因在于：

1. 问题建模缺陷：生成对抗样本的搜索空间呈指数级膨胀；
2. 奖励分配混乱：未实施操作最小化处理，导致奖励信号无法精准反馈核心操作；
3. 探索效率低下：在60次尝试限制下，无法有效探索庞大搜索空间并习得有效策略。