基于改进支持向量机的复杂软件分支漏洞智能预测

唐 玲[1*] 马 凌[1] 喻 幸[2]
TANG Ling MA Ling YU Xing

摘 要 复杂软件分支漏洞中存在随机发散的隐蔽特性，导致难以筛选关键信息。为此，文章提出了一种基于改进支持向量机的智能预测方法。建立面向复杂软件分支的智能模糊测试框架，在导向搜索模块，智能引导测试资源向高风险区域集中，通过特征选择与预处理关键字的方式，提取模块生成的测试用例，筛选出关键信息。在支持向量机分类关键信息中，通过约束关键信息参数保持最大值的方式实现改进，优化预测结果的输出。实验结果表明，所提方法在预测准确率与效率上均有显著提升。

关键词 改进支持向量机；软件分支结构；智能预测；漏洞检测；软件测试

doi：10.3969/j.issn.1672-9528.2025.06.028

0 引言

复杂软件系统中的分支漏洞，由于其隐蔽性强、利用难度低等特点，对信息系统的安全构成了严重威胁。因此，如何高效、准确地预测复杂软件中的分支漏洞，成为当前研究的热点和难点问题。

刘亚鹏[1]提出基于深度学习的开源软件安全漏洞预测方法。通过构建深度学习模型，对开源软件的历史漏洞数据进行学习，实现了对软件安全漏洞的有效预测。然而，该方法主要聚焦于开源软件的整体漏洞预测，对于复杂软件中的特定类型漏洞，如分支漏洞，其预测过程过拟合，精度有待提升。张珣等人[2]提出局部时空图卷积方法，并将其应用于网络漏洞预测。利用图卷积神经网络对漏洞数据中的拓扑结构和时序信息进行建模，输出漏洞预测结果。然而，该方法主要针对网络层面的漏洞进行预测，对于软件内部特别是复杂软件分支中的漏洞预测，其适用性仍需进一步验证。

鉴于现有研究的不足，本文提出一种基于改进支持向量机的复杂软件分支漏洞智能预测方法。

1 建立面向复杂软件分支的智能模糊测试框架

构建模糊测试框架，主要包括 4 个重要模块：静态分析模块、目标分支提取模块、关键字提取模块和导向搜索模块。4 个模块之间形成了一个闭环的测试流程：（1）从静态分析模块开始，收集程序信息；（2）通过目标分支提取模块筛选出需要测试的目标分支；（3）关键字提取模块进一步缩减输入空间；（4）导向搜索模块在约减的输入空间中生成测试用例并执行测试。随着测试的进行，新的半覆盖分支可能会被发现并反馈回流程的开始部分进行新一轮的分析和测试。整个框架通过不断迭代和优化，实现了对复杂软件分支漏洞的高效检测。

1.1 静态分析模块

静态分析模块包含分支提取和静态插桩两个功能。分支提取功能在编译时对程序中的条件分支相关信息进行收集，生成对应的分支导向信息[3]。该分支导向信息表征为六元组：

$$T=(i_{\mathrm{d}},m,v,w,e_{\mathrm{t}},e_{\mathrm{f}}) \tag{1}$$

式中：i_{d} 表示该分支的标签值，通过特定算法计算得到，用于唯一标识一个条件分支；m 表示该条件分支被引用的跳转条件符，包括等于、不等于、无符号数和有符号数的大于、大于等于、小于、小于等于；v 表示条件分支的目标操作数；w 表示该分支操作数的位宽；e_{t} 和 e_{f} 表示该分支所在基本块与两个后续分支之间的边的标签值。

静态插桩功能在源码中引入额外逻辑，以生成目标程序中所有条件分支的运行时信息[4]。使用影子位图存储条件分支操作数，以实时分析变异操作对条件分支操作数的影响。

1.2 目标分支提取模块

条件分支的覆盖状态可分为未覆盖、半覆盖和全覆盖 3 种。设 C 为条件分支的集合，对于任意 $c\in C$，其覆盖状态 S_c 可表示为：

$$S_c\in\{F_{\mathrm{w}},F_{\mathrm{b}},F_{\mathrm{q}}\} \tag{2}$$

1. 湖南信息学院 湖南长沙 410151
2. 湖南省高速公路交通警察局湘潭支队 湖南湘潭 411100
[基金项目] 2024 年度湖南省教育厅科学研究项目（24C1106）：基于遗传算法的复杂分支软件漏洞智能模糊测试框架构建及其测试用例优化方法研究

式中：F_w 表示未覆盖；F_b 表示半覆盖；F_q 表示全覆盖。未覆盖状态 F_w 指不存在测试用例能覆盖 c 的任意一个输出边（即 e_t 和 e_f）；半覆盖状态 F_b 指 c 的两个输出边中有且仅有一个被覆盖 [5]；全覆盖状态 F_q 则指两个出边均被覆盖。

通过分支覆盖率分析 e_t 和 e_f 的覆盖情况，筛选出半覆盖状态的条件分支作为目标分支 [6]。分支覆盖率可以通过公式计算为：

$$d = \frac{|\beta_c|}{|\beta_t|} \tag{3}$$

式中：β_c 表示被覆盖的分支集合；β_t 表示总分支集合。

1.3 关键字提取模块

针对目标程序中存在的半覆盖状态 F_b 对应的条件分支，识别出与该分支相关的输入字节可以有效缩减需要探索的输入空间。

基于操作数感知的轻量级关键字段识别方法，通过对变异前后条件分支操作数变化的分析，快速识别目标分支相关的输入字节 [7]。设 I 为输入字节的集合，对于任意 $i \in I$，其与目标分支 c 的相关性 $R_{i,c}$ 可表示为：

$$R_{i,c} = \begin{cases} 1, \text{if } i\text{与}c\text{相关} \\ 0, \text{otherwise} \end{cases} \tag{4}$$

通过公式（4）筛选出与目标分支相关的输入字节集合：

$$I_c = \{i \mid R_{i,c} = 1\} \tag{5}$$

将所有提取到的关键字汇总，得到输入字节集合，为后续漏洞预测提供依据。

1.4 导向搜索模块

基于提取目标分支和关键字段基础上，在约束的输入空间中引入改进支持向量机进行导向搜索，快速生成可突破目标分支的测试用例。一旦测试用例成功突破目标分支，即被加入至种子队列中，等待调度测试 [8]。

通过对半覆盖分支的不断迭代搜索，提升对目标程序的测试覆盖率，从而提升漏洞挖掘效率。通过 4 个模块的引入和详细描述，建立面向复杂软件分支的智能模糊测试框架，为复杂软件分支漏洞的检测提供了新的方法和技术手段。

2 基于改进支持向量机的漏洞预测方法

针对导向搜索模块，将模糊测试框架中关键字提取模块生成的测试用例和漏洞信息作为输入项，输入到导向搜索模块的改进支持向量机中，通过提取软件分支的特征，训练支持向量机模型，进行迭代训练分析，输出结果为预测的潜在漏洞区域。

2.1 关键信息特征选择与预处理

传统 SVM 通常直接使用原始特征，但易受到噪声影响，因此通过函数 Select 从原始特征集中选择对漏洞预测有用的特征子集。这一特征选择方法可以减少噪声特征的影响，提高模型的预测性能。根据软件分支的特点，在高度相关的输入字节集合 I_c 中选择了一系列特征，包括分支的复杂度、分支的调用频率、分支的修改历史等，构建了关键信息特征集 $\boldsymbol{F}$。特征选择可以表示为：

$$\boldsymbol{F}' = S(\boldsymbol{F}) \tag{6}$$

式中：S 函数用于从原始特征集中选择出对漏洞预测有用的特征子集 $\boldsymbol{F}'$。

特征预处理则是对特征进行标准化、归一化等操作，以提高模型的训练效果。设标准化后的特征集为 $\boldsymbol{F}''$，则：

$$\boldsymbol{F}'' = P(\boldsymbol{F}') \tag{7}$$

式中：P 函数用于对特征集进行预处理。

2.2 基于改进支持向量机的训练与参数优化

传统 SVM 的参数通常通过网格搜索或随机搜索进行优化，在面对复杂软件结构时，存在模型参数调整不够精确的问题。由此，本文通过约束关键信息参数保持最大值，实现关键信息参数优化，从而改进支持向量机，提升预测精度。具体过程为：基于改进支持向量机漏洞预测模型通过训练集进行训练，得到预测模型 [9]。设训练集为 D，训练过程可以表示为：

$$M = T(D, \boldsymbol{F}') \tag{8}$$

式中：T 函数用于根据训练集和特征集，训练得到预测模型 M。

为提高模型的预测性能，需要对模型的参数进行优化。设参数集为 θ，参数优化可以表示为：

$$\theta' = \max_\theta \left(\sum_{i=1}^{N} \lg P(y_i \mid x_i, \theta)\right) - \lambda \left\|\theta\right\|_2^2 \tag{9}$$

式中：N 表示训练集 D 中的样本数量；x_i 和 y_i 分别表示第 i 个样本的特征和标签；$P(y_i \mid x_i, \theta)$ 表示在给定参数 θ 下，模型预测 x_i 对应标签 y_i 的概率；$\max_\theta$ 表示使 $\sum_{i=1}^{N} \lg P(y_i \mid x_i, \theta)$ 取得最大值的 θ，实现了参数优化；$\|\theta\|_2^2$ 表示参数 θ 的二范数的平方和，用于防止过拟合；λ 表示正则化系数，用于平衡模型复杂度和训练误差。

最终，使用训练好的预测模型，对新的软件分支进行漏洞预测 [10]。设待预测的软件分支特征集为 $\boldsymbol{F}_{new}$，则预测结果可以用公式表示为：

$$P_r = s_g\left(\sum_{j=1}^{M} \alpha_j K(F_{new}, F_j) + b\right) \tag{10}$$

式中：M 表示支持向量（即训练集中对模型决策边界有影响的样本）的数量；α_j 表示第 j 个支持向量的权重；$K(\boldsymbol{F}_{new}, \boldsymbol{F}_j)$ 表示待预测特征集 $\boldsymbol{F}_{new}$ 和第 j 个支持向量特征集 $\boldsymbol{F}_j$ 之间的核函数，用于计算两者之间的相似度；b 表示模型的偏置项。s_g 函数用于将模型的输出转换为分类结果（正数表示存在漏洞，

负数表示不存在漏洞）。

模糊测试框架和漏洞预测模型可以协同工作，形成一个闭环的漏洞发现与修复流程。首先，利用模糊测试框架预测潜在漏洞区域，生成测试用例和漏洞信息；然后，对区域进行测试，触发并捕获漏洞。由此，有助于软件工作人员根据测试结果进行漏洞修复和代码优化。

3 对比实验

3.1 实验数据

采用美国宇航局开发的 MDP（metric data program）系统，挑选 6 个具有代表性的样本，分别是 CM1、JM1、KC1、MC2、PC1 以及 PC5。在软件工程或其子系统的背景下，每个选定的模块均与一组特定的数据集相关联，这些数据集不仅涵盖了 N 个程序代码度量元属性值，还包含一个至关重要的分类标签——Defective（即瑕疵存在与否的标识，具体取值为 Y 或 N），用于指示模块是否存在漏洞。其中，软件测试漏洞爬取部分代码如图 1 所示。

```
BuildPharaBnwOx := func(Stub string, SerializationByte []byte, hashModel int) []byte {
    stubByte := []byte(Stub)
    metaDataLen := make([]byte, 4)
    binary.LittleEndian.PutUint32(metaDataLen, uint32(len(SerializationByte)))
    manifestMid := []byte{1, 0, 0, 0, 17, 0, 0, 0, 1, 0, 0, 0, 0, 0}
    fileX := []byte{8, 0, 0, 0, 116, 101, 115, 116, 46, 116, 120, 116, 4, 0, 0, 0, 103, 114, 179,
    flag := []byte{0, 0, 0, 71, 66, 77, 66}
    // 数据总长度
    dataLen := make([]byte, 4)
    binary.LittleEndian.PutUint32(dataLen, uint32(len(SerializationByte)+len(manifestMid)+8))

    pharBuffer := bytes.NewBuffer([]byte{})
    pharBuffer.Write(stubByte)
    pharBuffer.Write(dataLen)
    pharBuffer.Write(manifestMid)
    pharBuffer.Write(metaDataLen)
```

图 1 软件测试漏洞爬取部分代码

利用本文提出的方法进行复杂软件分支漏洞预测，得到部分漏洞报告结果如图 2 所示。

```
at java.desktop/javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:1972)
at java.desktop/javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2313)
at java.desktop/javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:405)
at java.desktop/javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:262)
at java.desktop/javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:279)
at java.desktop/java.awt.Component.processMouseEvent(Component.java:6626)
at java.desktop/javax.swing.JComponent.processMouseEvent(JComponent.java:3389)
at java.desktop/java.awt.Component.processEvent(Component.java:6391)
at java.desktop/java.awt.Container.processEvent(Container.java:2266)
at java.desktop/java.awt.Component.dispatchEventImpl(Component.java:5001)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2324)
at java.desktop/java.awt.Component.dispatchEvent(Component.java:4833)
at java.desktop/java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4948)
at java.desktop/java.awt.LightweightDispatcher.processMouseEvent(Container.java:4575)
at java.desktop/java.awt.LightweightDispatcher.dispatchEvent(Container.java:4516)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2310)
```

图 2 部分漏洞报告结果

通过图 1~2 可以看出，所提方法可以有效实现复杂软件分支漏洞的检测，获得实际的漏洞检测结果，具有有效性。

以所提方法为实验组、基于深度学习的预测方法为对照 A 组，基于局部时空图卷积的预测方法为对照 B 组，对 3 种预测方法进行应用性能评价。

3.2 最大适应值平均值对比

将 3 种预测方法的最大适应值平均值变化情况绘制如图 3 所示。实验数据显示，实验组方法稳定性和准确性更高，数据集取值变量数量显著影响预测模型性能，n=6 为最佳变量数量，预测效果最佳。

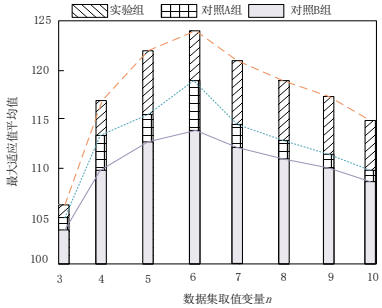

图 3 三种预测方法最大适应值平均值变化情况

3.3 收敛概率对比

在控制变量 n=6 的情况下，对比 3 种预测方法中对应算法的收敛概率，结果绘制如图 4 所示。

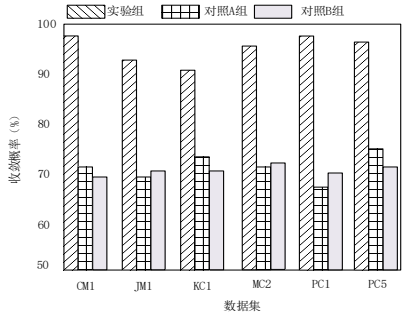

图 4 三种预测方法收敛概率对比结果

分析图 4，实验组的预测算法收敛概率持续保持在 90% 以上，两组对照组的预测方法的收敛概率普遍低于 75%。表明实验组的算法不仅能够更可靠地收敛到稳定的预测状态，而且在面对复杂多变的数据集时也能保持较高的预测准确性和稳定性。

3.4 平均收敛代数对比

平均收敛代数反映了预测模型在训练过程中的效率和速度。结果如图 5 所示。

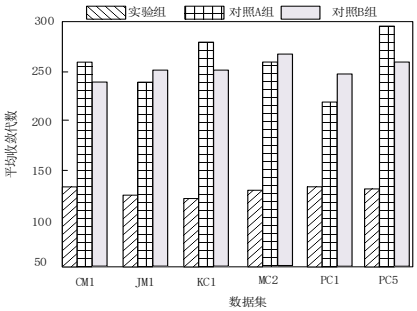

图 5 三种预测方法平均收敛代数对比图

实验结果显示，实验组预测方法平均收敛代数最低，表明其算法收敛效率高，所需迭代次数少，能快速稳定达到较优预测性能。这一优势在处理大规模数据集或需快速预测的场景中尤为关键，能大幅减少计算时间和资源消耗，提升任务执行效率。

（下转第 120 页）

[5] RONNEBERGER O, FISCHER P, BROX T. U-Net: convolutional networks for biomedical image segmentation[C]//Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Berlin: Springer, 2015: 234-241.

[6] LIU Z, HU H, LIN Y T, et al. Swin Transformer v2: scaling up capacity and resolution[C]//2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2022: 12009-12019.

[7] ZHANG X Q, LIANG T, TANG Z J, et al. Restoration of embedded image from corrupted stego image [J]. Signal processing, 2012, 92(7):1691-1698.

[8] VEERAKUMAR T, SUBUDHI B N, ESAKKIRAJAN S. Empirical mode decomposition and adaptive bilateral filter approach for impulse noise removal[J]. Expert systems with applications, 2019, 121: 18-27.

[9] NADEEM M, HUSSAIN A, MUNIR A, et al. Removal of random valued impulse noise from grayscale images using quadrant based spatially adaptive fuzzy filter[J]. Signal processing, 2020, 169:107403.

[10] DONG Y, ZHANG X Q, YU C Q, et al. Secret image restoration with convex hull and elite opposition-based learning strategy[J]. IEEE signal processing letters, 2023, 30: 195-199.

[11] ZHANG X Q, LI X L, TANG Z J, et al. Noise removal in embedded image with bit approximation[J]. IEEE transactions on knowledge and data engineering, 2020, 34(3): 1359-1369.

【作者简介】

文来（2002—），男，四川资阳人，本科在读，研究方向：机器学习、计算机科学与技术。

阴明旭（1998—），男，四川广安人，硕士研究生，助教，研究方向：机器学习、数字图像处理、智能驾驶感知。

（收稿日期：2024-12-31 修回日期：2025-05-09）

（上接第 115 页）

4 结语

本文围绕复杂软件分支漏洞的智能预测问题，提出了一种基于改进支持向量机的预测方法。该方法不仅提高了预测的准确率，还降低了计算复杂度，为软件安全漏洞的实时预测提供了有力支持。展望未来，加强跨领域合作，推动软件安全漏洞预测技术在实际软件系统中的应用落地，将是提升软件安全水平的关键所在。

参考文献：

[1] 刘亚鹏 . 基于深度学习的开源软件安全漏洞预测方法研究 [J]. 计算机应用文摘 , 2024, 40(11):110-112.

[2] 张珣 , 张楚童 , 艾孜孜·吐尔逊 , 等 . 一种局部时空图卷积方法及其在网络漏洞预测的应用 [J]. 科技导报 , 2023, 41(13):67-75.

[3] 鲁法明 , 唐梦凡 , 包云霞 , 等 . 目标导向的多线程程序 UAF 漏洞预测方法 [J]. 软件学报 , 2023, 34(7):3043-3063.

[4] 郑炜 , 刘程远 , 吴潇雪 , 等 . 基于知识图谱的跨项目安全缺陷报告预测方法 [J]. 软件学报 , 2024, 35(3):1257-1279.

[5] 胡雨涛 , 王溯远 , 吴月明 , 等 . 基于图神经网络的切片级漏洞检测及解释方法 [J]. 软件学报 , 2023, 34(6):2543-2561.

[6] 李彤彤 , 王诗蕊 , 张耀方 , 等 . 面向工控系统漏洞的多维属性评估 [J]. 计算机工程与科学 , 2023, 45(2):261-268.

[7] 罗宏芳 , 王春枝 . 基于贝叶斯攻击图的光网络安全态势预测研究 [J]. 激光杂志 , 2023, 44(8):134-138.

[8] 丁朝晖 , 张伟 , 杨国玉 , 等 . 工业控制系统网络攻击预测技术研究 [J]. 电子技术应用 , 2023, 49(1):86-90.

[9] 李秋月 , 韩道军 , 张磊 , 等 . 基于分层注意力网络和积分梯度的细粒度漏洞检测方法 [J]. 计算机科学 , 2024, 51(12): 326-333.

[10] 张俊 , 李山山 , 李磊 , 等 . 基于残差门控图卷积网络的源代码漏洞检测 [J]. 计算机工程与应用 , 2023, 59(22):293-299.

【作者简介】

唐玲（1984—），女，湖南长沙人，硕士，讲师，研究方向：软件测试、模糊测试。

马凌（1981—），女，吉林乾安人，硕士，教授，研究方向：计算机应用技术。

喻幸（1981—），男，湖北汉川人，硕士，高级工程师，研究方向：信息技术应用。

（收稿日期：2025-02-21 修回日期：2025-06-11）