基于时间因子优化的低空物联网第三方库漏洞轻量化识别方法

高燕[1]，罗琴[2]
（1. 中国电子科技集团公司第三十研究所，四川 成都 610000；2. 西南石油大学计算机与软件学院，四川 成都 610000）

摘　要：低空物联网的通信、导航等核心功能高度依赖第三方库，而此类库中的漏洞可能引发无人机失控、数据泄露等重大安全风险。针对现有漏洞识别方法难以及时捕捉新迁移库漏洞，且难以在资源受限的物联网设备中高效运行等问题，提出了一种基于时间因子优化的迁移库漏洞识别方法。该方法通过深度挖掘开源项目中的迁移信息，构建包括时间支持度、标签支持度在内的 6 类指标，筛选出新颖的轻量级迁移库。在此基础上，采用精简 Transformer 模型对所选库进行漏洞检测，降低边缘设备的计算负担，实现对漏洞的轻量化准确识别。实验结果表明，所提方法在漏洞识别任务中的 *F*1 值平均达到 0.78，相比主流方法提升 10% 以上，训练时间缩短了约 58%，平均预测时间仅 4.7 ms，能够有效提升低空场景下库迁移过程的安全性与实时性，为低空物联网设备提供高效的安全防护。

关键词：低空物联网；第三方库；漏洞识别；时间因子
中图分类号：TN915.08
文献标志码：A
doi：10.11959/j.issn.2096−3750.2025.00519

Time factor-optimized lightweight identification method for third-party library vulnerabilities in low-altitude IoT

GAO Yan[1], LUO Qin[2]
1. The 30th Research Institute of China Electronics Technology Group Corporation, Chengdu 610000, China
2. School of Computer Science and Software Engineering, Southwest Petroleum University, Chengdu 610000, China

Abstract: The core functions of low-altitude Internet of things (IoT), such as communication and navigation heavily rely on third-party libraries. Vulnerabilities in third-party libraries can lead to significant risks such as drone loss of control and data leakage. To address the limitations of existing vulnerability identification methods, such as difficul-ties in promptly detecting vulnerabilities in newly migrated libraries and inefficiencies when running on resource-constrained IoT devices, a migration library vulnerability identification method based on time factor optimization was proposed. By deeply mining migration information from open-source projects, six metrics, including temporal support and label support, were constructed to screen novel and lightweight migration libraries. A streamlined transformer model was employed to detect vulnerabilities in the selected libraries, which reduced the computational burden on edge devices and enabled light-weight yet accurate vulnerability identification. Experimental results demonstrated that the proposed method achieved an average *F*1-score of 0.78 in vulnerability identification tasks, outperforming mainstream approaches by more than 10%. Training time was reduced by approximately 58%, and the average prediction time was only 4.7 ms. The method effectively enhanced both the security and real-time performance of library migration in low-altitude scenarios, providing efficient protection for low-altitude IoT devices.

Key words: low-altitude IoT, third-party libraries, vulnerability identification, time factor

收稿日期：2025−07−20；**修回日期：**2025−09−12

0 引言

随着低空物联网设备（如无人机、高空基站等）的快速发展与广泛应用，复用第三方库已成为开发者的共识。借助 PlatformIO、Dronecode 等轻量化库托管平台及 github、git 等开源平台，开发者可快速集成通信协议库（如 MAVLink）、定位库（如 PX4-GPS）等关键组件，并实现各类应用功能的快速迭代。然而，这些第三方库在设备生命周期中可能面临漏洞暴露、协议过时或硬件兼容性失效等问题，其安全性动态变化迫使开发者频繁寻找安全替代库并执行更新[1]。受限于低空设备的计算资源与能耗约束，低空物联网第三方库面临如下挑战：（1）低空场景特有的新型攻击（如 GPS 欺骗、链路劫持）频发。2023 年无人机 CVE（common vulnerabilities and exposures）漏洞数量同比增长 67%，而早期库（如 2016 年前发布的版本）中高危漏洞占比高达 34%。传统的迁移推荐方法仍依赖历史迁移频率，无法捕获近期披露的高危漏洞库，导致新型威胁难以及时发现并处置。（2）开源社区涌现的海量候选库（如通信协议库 MavLink、定位库 PX4-GPS）规模呈爆炸式增长（如 Maven 仓库中的 Java 库数量已超过 50 万），而低空设备端计算资源受限（如 STM32F4 仅具备 1 MB Flash 存储空间），导致漏洞识别模型训练与预测耗时显著增加，难以满足实时性要求。

库迁移的时效性筛选与资源受限环境下漏洞的轻量化检测，是第三方库安全应用的关键，现有研究主要围绕库迁移推荐与漏洞检测模型展开，具体包括基于历史迁移频率的方法，如 Teyton 等[2]通过挖掘依赖变更记录生成迁移规则，并依据迁移频次阈值对候选库进行筛选。该方法虽简单有效，但数据源单一且未充分考虑时间因素，难以应对快速演化的漏洞威胁。基于多维度上下文与知识图谱的方法，如 He 等[3]提出的多指标排序算法，Li 等[4]构建的项目–库关联知识图谱。这些方法虽提升了推荐的置信度，但其核心仍依赖于历史数据，无法优先筛选近期修复的高危漏洞库。基于深度学习的漏洞检测方法，如 LightXML 等模型在极端多标签分类任务上表现卓越，但其庞大的计算开销难以部署于 STM32 等资源受限的低空物联网设备[5]。为缓解对标注数据的依赖，部分研究引入零样本学习框架：Lyu 等[6]利用词频–逆文档频率建模漏洞报告与标签的相关性；Tang 等[7]提出一种无数据零样本学习方法，利用 CLIP 特征恢复虚拟样本，避免训练数据隐私泄露；Zhang 等[8]提出了组合零样本学习方法，通过“编码–重排–解码”机制生成合成特征，以提升模型在开放世界中的泛化能力。此外，产业界也在积极探索低空设备的安全防护方案。例如，信安世纪提出了低空无人机数据全生命周期加密方案，结合国密算法保障“端–边–云”信道安全；绿盟科技开发了无人机攻防靶场，系统挖掘并复现了九大类典型漏洞。综上所述，现有方法普遍存在以下局限：过度依赖历史统计量，缺乏对漏洞时效性的动态感知机制，导致推荐结果偏向陈旧版本库，无法有效应对低空场景中因技术迭代快、新型攻击频发所带来的动态安全威胁。同时，高性能漏洞检测模型与终端设备的资源约束之间的矛盾依然突出。因此，亟须发展一种融合时间因子优化与轻量化检测的漏洞识别方法，以实现对高危漏洞库的实时、精准、高效识别。

在此背景下，本文提出了基于时间因子优化的低空物联网第三方库漏洞轻量化识别方法。首先，重构时间因子优化推荐算法，将传统以迁移频次为核心的置信度计算升级为漏洞时效驱动模型。基于动态时间窗口（如近 6 个月）量化库漏洞的披露时间与当前固件版本的间隔，利用时间因子对近期高危漏洞库进行加权提权，从而使安全替代库的推荐优先级与漏洞修复时效性紧密关联。其次，针对地勘设备端的资源约束（如 STM32 的 1 MB 内存限制），采用深度模型轻量化技术，在保留 Transformer 特征提取能力的前提下，通过编码器层压缩以降低参数规模，并引入知识蒸馏机制，以原始漏洞识别模型作为教师网络，蒸馏出参数量来显著减少学生模型，从而实现资源受限环境下的实时漏洞检测。最后，利用时间因子优化推荐算法筛选出高时效性候选库，再由轻量化引擎在资源受限环境下完成精准识别，两者协作，实现低空场景下第三方库漏洞的有效识别。需要指出的是，尽管低空物联网第三方库更新频率比通用软件高，但固件版本的同步周期通常较长，导致现有漏洞检测方法在实际应用中可能面临时效性不足的挑战。本文方法将重点聚焦于第三方库的漏洞识别，为软件全生命周期尤其是开发阶段提供安全的第三方库，进而减少低

空物联网发布软件的漏洞，缓解因固件版本同步之后所带来的安全风险。

本文的主要贡献如下。

1) 提出了时间因子优化推荐算法，重构以迁移频次为核心的置信度计算模型，引入基于动态时间窗口的时间支持度指标，使推荐优先级与漏洞修复时效性强关联。

2) 提出了面向低空物联网的轻量化Transformer与蒸馏架构，设计了参数更少的轻量化Transformer编码器，并引入知识蒸馏机制，以经时间因子筛选后数据训练的模型为教师，蒸馏出兼具高精度与低耗能的学生模型。

3) 提出了融合时间因子优化推荐算法的迁移库漏洞识别VLTA-LA（vulnerability libraries detection with time-aware for low-altitude IoT scenarios），通过时间因子优化推荐算法筛选出高时效性候选库，再由轻量化引擎在资源受限环境下完成精准、快速的漏洞识别。

1　VLTA-LA整体框架

在整体框架原理展开前，先对本方法原理中用到的术语进行说明：“轻量化”指漏洞检测模型的参数量小于10^6，且模型推理时间低于10 ms，满足IEEE 802.15标准；“时间因子优化”指动态调节时效性权重的算法机制。

基于上述概念，本文提出的VLTA-LA整体框架如图1所示。该框架包括以下3个阶段：第一阶段，第三方库数据采集。VLTA-LA从无人机开源社区（如Dronecode）、物联网CVE数据库、GitHub等平台挖掘开源项目相关信息，并分析库迁移的实际案例。同时，系统从Maven中央仓库等渠道获取第三方库的最新数据，进行统一解析与集成，为后续的分析提供全面、多元的数据基础。第二阶段，时间感知库筛选。VLTA-LA设计时间支持度指标，并构建时间因子优化的库筛选算法，输出高危漏洞库优先级列表，从而筛选出更具时效性的迁移第三方库。第三阶段，漏洞识别与评估。VLTA-LA使用基于深度学习的轻量化漏洞识别引擎对筛选库进行漏洞识别，并分类评估模型的漏洞识别效果。

将低空物联网设备固件仓库、GitHub等仓库视为一个独立的项目，同时为了简化依赖检索过程，以托管在Maven平台上的项目为核心，对项目集$\boldsymbol{P}$中的每个项目，通过解析它的Dockerfile及pom.xml等配置文件来检索依赖项，构造三元组<G:A:V>，通过这种方式，VLTA-LA能够系统地识别每个项目所依赖的第三方库并生成库集合$\boldsymbol{L}$。之后，VLTA-LA对GitHub项目的提交日志进行解析，从而挖掘迁移时间戳及依赖项变化。对于某个项目p，若提交c_{i-1}先于c_i，就定义c_{i-1}为c_i的父提交。p在c_{i-1}时的依赖项集合为$\boldsymbol{L}_{c_{i-1}}$，p在c_i时的依赖项集合为$\boldsymbol{L}_{c_i}$，通过比较$\boldsymbol{L}_{c_{i-1}}$和$\boldsymbol{L}_{c_i}$可以生成依赖变化项，该过程由以下表达式定义

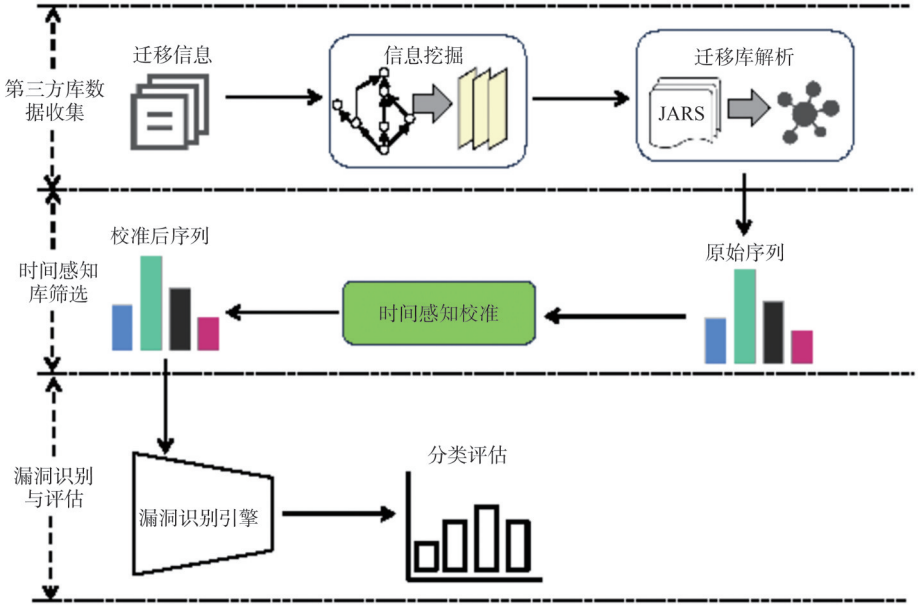


图1　VLTA-LA整体框架

$$\boldsymbol{D}_{c_i}^{+} = \boldsymbol{L}_{c_i} - \boldsymbol{L}_{c_{i-1}} \tag{1}$$

$$\boldsymbol{D}_{c_i}^{-} = \boldsymbol{L}_{c_{i-1}} - \boldsymbol{L}_{c_i} \tag{2}$$

其中，$\boldsymbol{D}_{c_i}^{+}$和$\boldsymbol{D}_{c_i}^{-}$分别表示项目p在c_i时添加和删除的依赖项集合。

通过收集项目集$\boldsymbol{P}$中所有项目在不同提交日志中添加和删除的依赖项集合，并进行笛卡尔积运算，可以得到迁移规则集合$\boldsymbol{M}$，随后，VLTA-LA将集合$\boldsymbol{M}$中的每个迁移规则与其对应的时间戳进行匹配，从而生成一个时间字典TD，其中每个时间点都关联着该时刻发生的所有不同迁移规则，其格式为< date, s, t >。

本文采用时间因子作为筛选机制，主要基于统计观察到的低空物联网漏洞特征分布收敛规律，通过对近180天披露的高危漏洞库进行统计分析（基于NVD数据集），发现其漏洞特征呈现收敛趋势Top5漏洞类型占比达到68%，应用程序编程接口（API, application programming interface）调用重复度增长86.8%，这种分布集中化现象源于低空物联网领域新型攻击（如GPS欺骗、链路劫持）的快速迭代，导致短期内漏洞模式呈现高度同质化。由此可见，定量分析时间支持度（TS, time support）筛选出的“高时效性库”的特征将对模型复杂度及知识蒸馏等过程至关重要。

2　基于时间因子优化的漏洞检测算法模型

2.1　迁移趋势图构建

随着时间的不断推移，第三方库既可作为源库被其他项目引用，也可作为目标库接收其他项目的迁移，即第三方库的生命周期通常会经历引入期（新库被广泛采用）、稳定期（维护性更新为主）和淘汰期（被更安全的替代库取代）3个阶段。在低空物联网环境下，第三方库迁移行为的时间动态特性更为明显。为了捕捉第三方库的迁移趋势与演化规律，VLTA-LA深入挖掘了不同提交日志中的迁移时间戳及其对应的候补迁移规则，并引入迁移趋势图（MTG, migration trend graph）算法，用带权有向图$G = (\boldsymbol{V}, \boldsymbol{E}, W)$表示。其中，$\boldsymbol{V}$是顶点集，表示三方库；$\boldsymbol{E}$是边集合，表示迁移事件；$W$是权重函数，$W_t$表示时间$t$发生的迁移次数。MTG算法通过量化分析开源项目提交日志中的依赖变更事件，建立了时间维度（提交时间戳）与行为维度（迁入/迁出）库迁移行为模型。其中，迁入事件表示漏洞修复后引入的安全库，迁出事件表因漏洞暴露被移除的风险库。MTG算法伪代码如算法1所示。

算法1　MTG算法

输入　提交日志集合$\boldsymbol{C}$，时间衰减因子τ，固件发布日期t_n

输出　迁移趋势图

```
1: MTG ← InitGraph() # 初始化空图
2: for c in C:
3:   if IsMigration():
4:     Δt ← |c.timestamp − t_n|
5:     decay ← exp(−Δt /τ)
6:     removed, added ← GetDependency（c）
7:     for (lib_out, lib_in) in CP(removed, added):
8:       UpdateNode(MTG, lib_out, “out”)
9:       UpdateNode(MTG, lib_in, “in”)
10:      UpdateEdge(MTG, lib_out, lib_in, decay)
11: return MTG
```

其中，1～3通过提交消息关键词（如“migrate”/“update”）识别库迁移操作；4～5行利用时间衰减计算确保近期时间越近权重越高；6行依赖变更分析明确库行为（迁入或迁出）；7～10行实现MTG图更新，包括迁入迁出次数及迁移路径等。

库迁移趋势的简化示例见表1。其中，采用构件ID来标识库，年份作为横轴，第三方库作为纵轴（包括通用的第三方库及MAVLink等低空物联网专用库）；上三角形符号用于标识库的引入（迁入）事件，即漏洞修复引入事件；而下三角形符号则代表库的移除（迁出）事件，即漏洞暴露移除事件；每个符号旁边附带的数字代表迁入迁出的数量。由表1可知，asm库在2006—2012年，以迁出为主，表明该库进入了淘汰期；2016—2020年重新进入引入期，被作为目标库被其他三方库引用；2017年被引入5次，这可能与该库2015年发布了新版本，并对已知漏洞进行了修复有关。antlr库呈持续迁出状态，表明该库不再适合作为依赖选择，可能存在架构缺陷。ars4j库与asm库类似，先进入淘汰期，近年来重新进入引入期；MAVLink库在2016年后的迁入量快速增加，这与低空经济发展及无人机协议标准同步。

MTG通过将历史迁移事件与漏洞披露时序进行关联，使迁出事件峰值能够作为预测高危漏洞出现的关键指标。该方法通过构建迁移趋势图，直观

表1　库迁移趋势的简化示例

第三方库	2006年	2007年	2008年	2009年	2010年	2011年	2012年	2013年	2014年	2015年	2016年	2017年	2018年	2019年	2020年
asm	▼8	▼2	▼7	▼3	▲1	▲1▼14	▼16	▲4▼1	/	▼3	▲3	▲5	▲3	▲5▼3	/
antlr	/	/	▼1	/	▼1	/	/	▼3	▼1	▼16		▼4	▼5	/	▼6
args4j	/	/	/	/	/	▼1	/	▲1▼9	/	/	▼2	▼1	▲9	/	/
MAVLink	/	/	/	/	/	/	▲2	/	/	/	▲8	▼1	/	/	▲6

呈现迁移库的生命周期和流行度变化，从而捕捉不同时间段内各库的采纳和淘汰情况，为后续推荐算法的构建提供量化依据。在推荐阶段，迁移优先级由迁入与迁出权重共同决定，当迁入权重/迁出权重大于1.5时，即可将该库判定为安全候选库。

2.2　时间因子优化推荐算法

VLTA-LA的时间因子优化推荐算法如算法2所示。

算法2　时间因子优化推荐算法

输入　项目集$\boldsymbol{P}$，库集$\boldsymbol{L}$，标签库$\boldsymbol{D}$

输出　按$c_{\text{onf}}(s,t)$排序的列表

1: **for** $p \in \boldsymbol{P}$ **do**

2:　$\boldsymbol{D}_{c_i}^{+} \leftarrow (\boldsymbol{L}_{c_i} - \boldsymbol{L}_{c_{i-1}}) \cap \boldsymbol{L}$

3:　$\boldsymbol{D}_{c_i}^{-} \leftarrow (\boldsymbol{L}_{c_{i-1}} - \boldsymbol{L}_{c_i}) \cap \boldsymbol{L}$

4:　$M \leftarrow (\boldsymbol{D}_{c_i}^{+} \times \boldsymbol{D}_{c_i}^{-}) \cup M$

5:　$T_D \leftarrow M \cup <\text{date}, s, t>$

6: **end for**

7: **for** $<\text{date}, s, t> \in \mathcal{LD}$ **do**

8:　$Hc_{\text{onf}}(s,t) \leftarrow R_S(s,t) \cdot M_S(s,t) \cdot D_S(s,t) \cdot A_S(s,t)$

9:　$T_S(s,t) \leftarrow T_F / \min(\text{MaxValue}, T_G(s,t))$

10:　$c_{\text{onf}}(s,t) \leftarrow T_S(s,t) \cdot L_S(s,t) \cdot Hc_{\text{onf}}(s,t)$

11: **end for**

12 :**return** 按$c_{\text{onf}}(s,t)$排序的列表

针对项目集$\boldsymbol{P}$中的每个项目p，VLTA-LA先按其提交时间顺序对所有提交日志进行排序。在遍历这些提交日志时，系统将每一个提交c_i与其父提交c_{i-1}所依赖的第三方库进行比较，识别两者之间的差异，并基于上述差异生成候选的迁移规则。随后，通过引用时间字典TD中记录的迁移时间戳及其对应的迁移规则，计算每个迁移规则的时间支持度及库依赖随时间的迁移模式。

为量化迁移库的标签相关性，算法设计了标签支持度。

$$L_S(s,t) = \max(1, L_C(s,t) \cdot L_F) \tag{3}$$

其中，L_C表示源库和目标库之间的相同标签数量，L_F是一个可自定义的标签权重值。

为量化源库迁移至目标库的迁移频率，算法设计了规则支持度

$$R_S(s,t) = \frac{R_C(s,t)}{\max_{(s,x) \in \boldsymbol{R}_c} R_C(s,x)} \tag{4}$$

其中，$R_C(s,t)$表示迁移规则$<s,t>$发生的迁移次数。

为挖掘项目提交日志的隐藏迁移描述信息，算法设计了消息支持度

$$M_S(s,t) = \text{lb}(M_C(s,t) + 1) \tag{5}$$

其中，$M_S(s,t)$表示源库s迁移至目标库t的日志消息条数。

为实现多项目跨版本的约束优化，算法设计了距离支持度

$$D_S(s,t) = \frac{1}{R_C(s,t)} \sum \frac{1}{(\text{dis}(c_i, c_j) + 1)^2} \tag{6}$$

其中，$\text{dis}(c_i, c_j)$表示提交c_i和c_j的版本距离，如果提交发生在同一版本，则距离为0。

为量化源库和迁移库之间的真实代码修改次数，算法设计了API支持度

$$A_S(s,t) = \max(0.1, \frac{A_C(s,t)}{\max_{(s,x) \in \boldsymbol{R}_c} A_C(s,x)}) \tag{7}$$

为量化迁移规则的时效性，设计了时间支持度

$$T_S(s,t) = \frac{T_F}{\min(\text{MaxValue}, T_G(s,t))} \tag{8}$$

$$T_F = \lambda \cdot \exp(-\frac{|t_m - t_n|}{\tau}) \tag{9}$$

其中，λ表示时间权重因子，用于控制时间支持度在整体置信度中的权重；t_n表示迁移事件时间戳，即库迁移操作发生的具体时间；t_m表示当前固件的发布日期，即设备固件的发布版本时间；τ为漏洞半衰期，根据NVD漏洞生命周期统计结果默认设置为6个月；时间间隔$T_G(s,t)$表示迁移规则$<s,t>$的迁移时间与固件设备发布时间常量之间的绝对值，其值越小，说明迁移发生的时间越近，对该迁移的信赖度就越高。由于笛卡儿积生成的某些

迁移规则可能在实际开发中不存在，可能无法计算时间间隔。

通过使用以上6项指标，将项目信息、标签信息以及时间信息纳入推荐算法，并计算出最终置信度 c_{onf}。根据 c_{onf} 将候选迁移规则从高到低进行排序，筛选具有时效性的最佳迁移库。

Chronos方法主要基于漏洞报告，结合漏洞披露时间进行漏洞库识别，该方法主要依赖漏洞披露时间之间的静态关联[8]；而VLTA-LA通过迁移趋势图对第三方库生命周期进行动态建模，并结合动态时间窗口来量化迁移行为与漏洞修复之间的时效性关联，从而能够有效应对低空领域因技术快速迭代所导致的历史依赖滞后问题。两者的方法原理对比见表2。

表2　Chronos和VLTA-LA的方法原理对比

对比维度	Chronos	VLTA-LA
时间要素建模	漏洞披露时间	迁移事件时间戳+漏洞修复时效性
关键方法	TF-IDF加权	时间支持度(TS)+动态时间窗口
应用场景	通用软件	低空物联网库等快速迭代场景

2.3　漏洞识别模型

本文提出了一种基于时间因子优化的轻量化第三方库漏洞识别模型，其整体流程如图3所示。该模型创新地将Transformer架构与时间因子优化的推荐结果进行深度融合。具体而言，先通过第2.2节所述的时间因子优化推荐算法筛选出高时效性候选库，完整LightXML模型在此数据上训练得到的网络作为教师网络，借助知识蒸馏技术，将其中蕴含的“时效性知识”迁移至结构更精简的学生网络中，最终实现在资源受限终端上兼顾轻量化与高精度的漏洞识别。在模型结构方面，轻量化Transformer组件的具体参数见表3：编码器层数设为2层，词嵌入维度设为128，前馈网络维度设为512。实验表明，该配置能够在保证对时间感知筛选出的高时效性库漏洞特征捕捉能力的前提下，降低模型参数量和计算量，满足资源受限设备的部署需求。

在知识蒸馏阶段，温度参数 T 设置为2.0，以软化教师模型的预测分布，增强类别间关系的信息量。蒸馏损失权重 β 设为0.5，表示任务损失（二元交叉熵）与蒸馏损失（KL散度）在总损失函数中所占权重相当，二者贡献相当。教师模型（LightXML）采用与VLTA-LA相同的训练数据，即经由

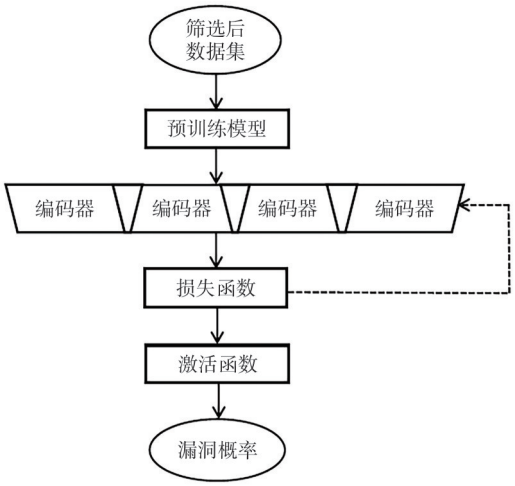


图2　模型整体流程

时间感知推荐算法筛选出的高时效性库，从而确保知识传递过程中的上下文一致性。

模型训练采用AdamW优化器，基础学习率设为 1×10^{-5}，并应用带线性预热（前10%训练步数）和线性衰减的学习率调度策略。权重衰减设为0.01，以抑制过拟合。轻量化学生模型训练轮数设为50轮，并启用早停策略（当验证损失连续5轮不下降时则停止训练），因其收敛速度快于教师模型（150轮）。

表3　模型参数

机制	参数	普通模型（教师）	轻量化模型（学生）
Transformer	编码器层数	4	2
	词嵌入维度	256	128
	前馈网络维度	1 024	512
	注意力头数	8	8
	注意力Dropout率	0.1	0.1
	优化器	AdamW	AdamW
	学习率	1×10^{-5}	1×10^{-5}
	权重衰减	0.01	0.01
	批量大小	32	32
	训练轮次	150	50
知识蒸馏	温度参数(T)		2.0
	蒸馏损失权重		0.5
	蒸馏损失类型		KL散度
输入/嵌入	词嵌入是否微调	是	是

模型的核心逻辑如图3所示。当预训练模型对文本编码生成特征向量后，首先经过注意力函数处理，将一个查询和一组键值对映射到一个输出。其中，查询、键、值和输出都是向量。然后，将输出

作为softmax函数的输入，得到注意力权重的加权和。采用缩放点积注意力作为注意力函数，并采用批量计算方式提高计算效率。其注意力函数定义为

$$\text{softmax}\left(\frac{\boldsymbol{QK}^{t}}{\sqrt{d}}\right)\boldsymbol{v}\in\mathbf{R}^{n\times v} \tag{9}$$

其中，$\boldsymbol{Q}\in\mathbf{R}^{n\times d}$表示维度为$d$的查询，$\boldsymbol{K}\in\mathbf{R}^{m\times d}$表示维度为$d$的键，$\boldsymbol{V}\in\mathbf{R}^{m\times v}$表示维度为$d$的值。

使用了多头注意力机制，使模型可以基于相同的注意力机制学习到不同的行为，再将不同的行为作为知识组合起来，以捕获序列内各种范围的依赖关系，如短距离依赖关系和长距离依赖关系。给定查询后，将不同注意力头的结果进行合并，得到注意力层的输出结果，并作为前馈网络的输入，用以捕捉和学习输入数据中的复杂关系，增强模型的表达能力。本文采用两个全连接层和Relu激活函数组成，其数学表达式为

$$\text{FFN}(x)=\max(0,xW_1+b_1)W_2+b_2 \tag{10}$$

其中，W_1和W_2均为权重参数，b_1和b_2为偏置参数。

特征向量经前馈网络处理后会被压缩成一维向量，最后在输出层通过sigmoid激活函数进行归一化处理，最终输出的文本属于各个标签的概率。

知识蒸馏过程主要包括训练教师模型、初始化学生模型与蒸馏训练。其中，训练教师模型采用完整的LightXML模型架构（配置见表3“普通模型（教师）”），在经时间感知推荐算法筛选后的训练库数据集上进行训练，优化目标为标准二元交叉熵损失函数。训练完成后固定其参数，作为教师模型使用。初始化学生模型，即构建轻量化Transformer模型（配置见表3“轻量化模型（学生）”）。蒸馏训练过程如下

1) 对于同一批训练数据，同时输入教师模型和学生模型进行前向传播。

2) 计算教师模型的原始logits输出z_t，并应用温度缩放得到软目标$p_t=\text{softmax}\left(\frac{z_t}{T}\right)$。

3) 计算学生模型的原始logits输出z_s，应用相同的温度缩放$p_s=\text{softmax}\left(\frac{z_s}{T}\right)$。

4) 通过任务损失与蒸馏损失加权形成损失计算，可表示为$l_{\text{total}}=\beta\cdot l_{\text{task}}+(1-\beta)\cdot l_{\text{distill}}$。其中，任务损失主要利用学生模型预测sigmoid(z_s)与真实二进制漏洞标签y之间的二元交叉熵损失实现，可表示为$l_{\text{task}}=\text{BCE}(\text{sigmoid}(z_s),y)$。蒸馏损失通过学生温度缩放$p_s$与教师软目标$p_t$之间的KL散度实现，可表示为$l_{\text{distill}}=\text{KL}(p_s\|p_t)$。

5) 计算学生模型参数的梯度，并使用AdamW优化器更新学生模型参数，实现反向传播与优化。在这个过程中仅更新学生模型参数，教师模型参数保持冻结。

6) 重复上述过程，直到达到预设的训练轮数（50轮）或触发早停条件（验证集损失连续5轮未下降）。

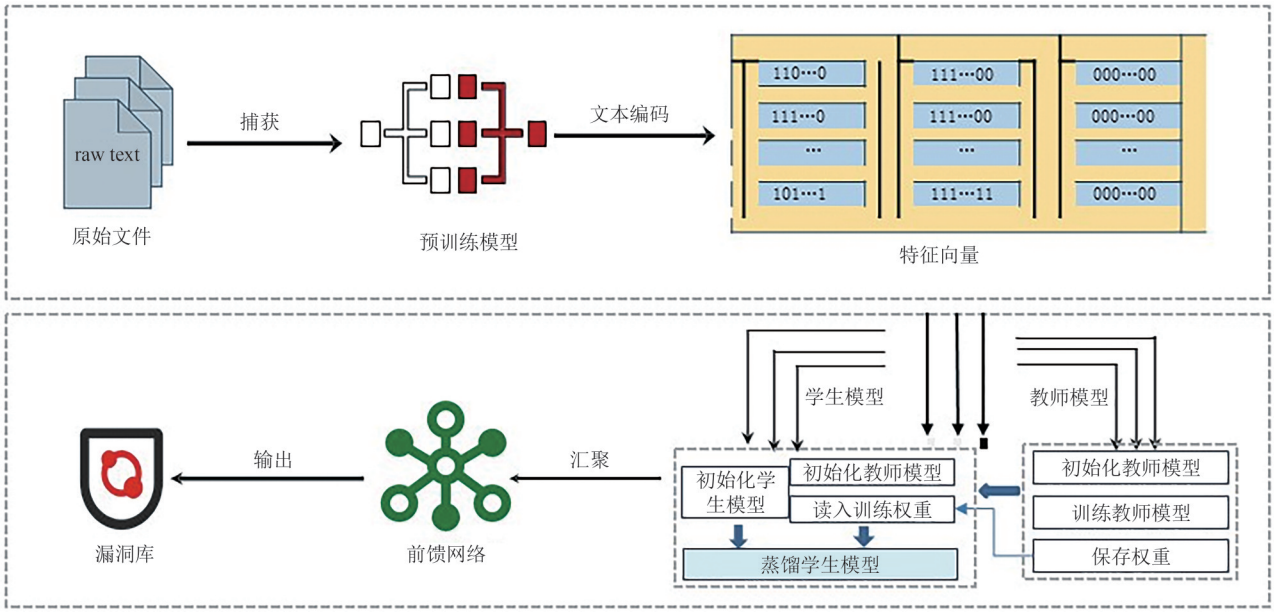


图3　模型核心逻辑

3　测试及分析

为验证本文所提模型效果，从以下 3 个方面进行验证：分析推荐算法的时效性；验证时间支持度占比对时间因子优化推荐算法性能的影响；将 VLTA-LA 与其他 3 种漏洞识别模型进行对比，评估 VLTA-LA 的漏洞识别效果。

3.1　数据集

实验使用了专用于物联网的第三方库漏洞库 IoTLib、无人机固件漏洞数据库 DroneSec、开源项目索引数据集 Libraries.io。其中，Libraries.io 包含来自 33 个包管理器的项目元数据。为了确保所选项目具有较高的质量，选择了那些拥有至少 10 颗星标且非分叉的项目，处理后形成 GT2014 数据集，其信息见表 4。

表 4　　GT2014 数据集信息

数据类型	数量
软件项目	19 652
迁移提交	25 245
迁移规则	10 611
第三方库标签	53 561

为验证 VLTA-LA 的漏洞识别能力，实验使用了文献[9]中的 NVD 漏洞数据集，其信息见表 5。

表 5　　NVD 漏洞数据集信息

名称	数量
NVD 漏洞实体	130 115
SCA 漏洞实体	74 664
标记漏洞实体	7 696
漏洞库	4 682

3.2　时效性分析

汇总了 2005—2020 年历年的迁移提交次数，用以探究迁移活动的时间分布，如图 4 所示。在 2012 年之前，迁移提交次数显示出较为缓慢的增长趋势，这可能是当时的软件开发社区规模较小，导致可选择的第三方库相对有限。从 2012 年的 1 378 次急剧增长到 2013 年的 2 672 次，数量增长接近一倍。虽然在 2014 年因特定因素导致迁移次数有所回落，但从总体趋势来看，增长速度在接下来的几年中仍然保持强劲，并且在 2016 年增长至 3 865 次，达到了峰值。然而，随着时间的继续推移，迁移次数呈现出逐步下降的趋势，并在 2020 年达到了 253 次的最低点。这可能是因为某些第三方库凭借其卓越的性能和稳定性在市场上确立了主导地位，赢得了开发者的广泛认可，使得他们的迁移需求有所降低。总体而言，迁移提交次数经历了先上升后下降的趋势。符合第三方库引入期、平稳期和淘汰期的阶段划分，但三个阶段可能会循环往复。

为验证时间因子优化推荐算法对更具时效性的迁移库的筛选能力，实验分别统计了 2016—2020 年的时间因子优化算法筛选前后迁移规则的年份占比分布，结果如图 6 所示。在筛选前，2016 年的迁移规则占比达到了 30.4%，占比位列第一名；而筛选后 2016 年的迁移规则占比为 23.62%，占比降低了

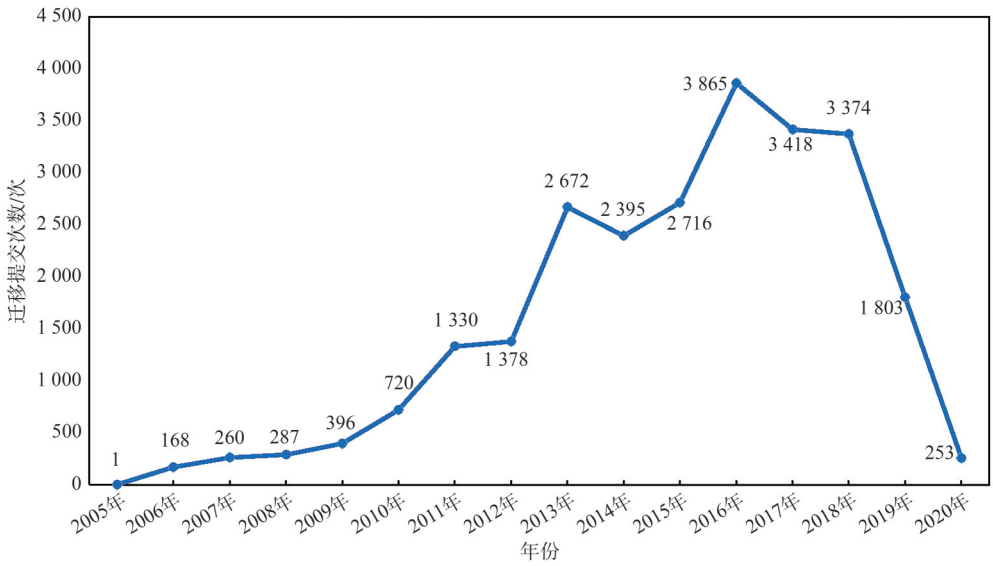
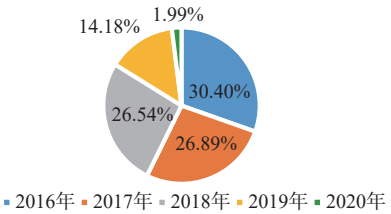
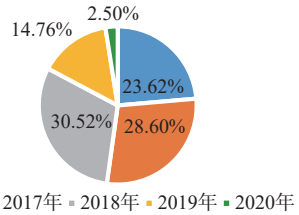


图 4　迁移提交次数

6.78%。同时，2018年的迁移规则占比由筛选前的26.54%上升至筛选后的30.52%。此外，2017年、2019年和2020年的迁移规则占比在筛选后分别上升了1.71%、0.58%和0.51%。以上结果表明，时间因子优化推荐算法能够很好地筛选出更加新颖和更具时效性的迁移第三方库。



（a）筛选前迁移规则年份占比



（b）筛选后迁移规则年份占比

图5　时间因子优化算法筛选前后迁移规则年份分布对比

3.3　性能评估

为衡量时间因子对推荐算法性能的影响，实验设置了不同水平的时间权重值，并引入了推荐系统中广泛使用的精确度（Precision）和召回率（Recall）指标进行评估。表达式如下

$$\text{Precision} = |R_k \cap R_t|/|R_k| \tag{11}$$

$$\text{Recall} = |R_k \cap R_t|/|R_t| \tag{12}$$

其中，R_k表示前k个推荐结果，R_t表示数据集中所有的真实迁移规则。

同时，为了综合考虑两个指标的特性，实验还使用了$F1$值，其表达式如下

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

时间因子对算法性能的影响如图6所示。关于精确度，在时间权重因子λ的值从100增加到400的过程中，VLTA-LA的精确度始终保持在0.835 1，未发生任何变化，这可能是因为在这个范围内λ的调整幅度不足以对推荐系统的精确度产生显著影响。然而，当λ进一步从400增长到500时，精确度显著提升，从0.835 1提高到0.850 5，达到最高值。此外，当λ设置为700时，VLTA-LA的精确度依然保持在0.850 5，与λ为500时相同。值得注意的是，在λ为700的两侧，精确度均出现了一定程度的下降。

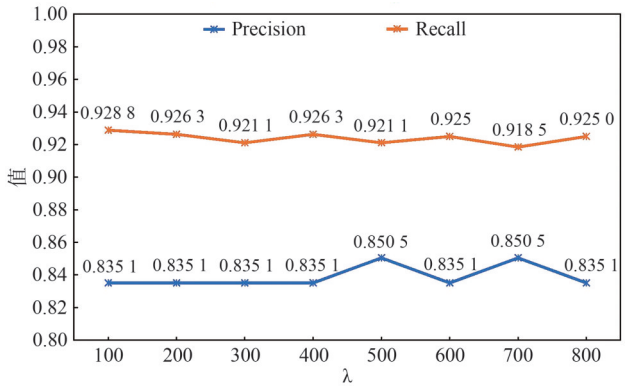


图6　时间因子对算法性能的影响

随着时间权重因子的增加，VLTA-LA的召回率呈现出非规律性波动。这表明召回率不仅受到时间权重调整的影响，还受到多种因素的综合作用，而λ为500时，VLTA-LA的精确度达到最优水平，且有着良好的召回率。

为了评估方法的推荐性能，实验将VLTA-LA与以下3种方法进行了比较

1) MMR[3]：设计了4种量化指标，从项目层面挖掘并排序迁移规则。

2) LMG[4]：通过构建库迁移图并分析迁移频率，推荐适用的第三方库。

3) MigrationMiner[10]：基于收集的特定代码片段，在方法层面推荐迁移库。

VLTA-LA与MMR、LMG、MigrationMiner的性能对比见表6，其中精确度、召回率和$F1$值的最高值均以黑体加粗突出显示。尽管MigrationMiner方法在精确度方面达到了最高值0.941 2，但其召回率仅为0.049 2，$F1$值仅为0.093 5，这两个指标显著偏低。这表明该方法虽然在推荐准确性上有出色表现，但在覆盖相关库方面存在明显不足。极低的召回率也限制了其在实际应用场景中的部署。VLTA-LA方法展现出了更为均衡的表现，召回率达到了最高值0.921 1，确保了广泛的覆盖率，同时$F1$值也达到了最高值0.884 4，这表明VLTA-LA方法在整体表现上能够达到最优水平。MMR和LMG也展现出了较为均衡的性能表现，但它们在精确度、召回率和$F1$值上的表现均不如VLTA-LA。

为进一步验证本文方法的有效性，设计了消融实验，其结果见表7。当移除时间因子后，漏洞识别

表 6　VLTA-LA 与 MMR、LMG、MigrationMiner 的性能对比

方法	精确度	召回率	*F*1 值
MMR	0.778 4	0.893 9	0.832 2
LMG	0.703 7	0.661 1	0.681 7
MigrationMiner	**0.941 2**	0.049 2	0.093 5
VLTA-LA	0.850 5	**0.921 1**	**0.884 4**

表 7　消融实验结果

方法	Precision@1	Recall@1	*F*1@1
非轻量化架构模型	0.88	0.66	0.75
移除时间因子	0.86	0.62	0.72
移除知识蒸馏	0.83	0.65	0.68
VLTA-LA	0.98	0.74	0.84

的精确度、召回率和 *F*1 值均出现了不同程度下降。其中，*F*1 值的影响最大，降低了 15% 左右。当移除知识蒸馏模块，本文方法在漏洞识别的指标与移除时间因子变化趋势类似。由此可见，本文方法在低空物联网环境下对第三方库漏洞识别的有效性。

3.4　漏洞识别效果评估

本实验将 VLTA-LA 与现有的 3 种漏洞识别模型进行对比，以评估其漏洞识别效果。为确保结果的公平性，所有模型均使用第 3.1 节所述的数据集进行训练和测试。数据集中正负样本比例在 1∶1.2 至 1∶1.8（不同数据集比例略有差异，同一数据集内比例保持一致），训练集、验证集及测试集的比例为 8∶1∶1，且时间戳覆盖率达到 90% 以上。

参与对比了 FastXML、Bonsai、LightXML 3 种模型。

FastXML[11]：一种基于树的分类器，使用交替最小化算法优化损失函数，能够在有限的迭代次数内实现收敛。

Bonsai[12]：基于树的分类器，利用浅层树学习算法，联合输入特征与输出标签信息进行多标签分类。

LightXML[13]：一种基于深度学习的分类器，通过动态负标签采样对 Transformer 模型进行微调，并借助生成式协作网络实现标签的召回和排序。

在评估指标方面，实验采用 Precision@*k*、Recall@*k* 和 *F*1@*k* 作为模型性能的评估指标。其中，Precision@*k* 指 *k* 个预测结果的精确度，Recall@*k* 指前 *k* 个预测结果的召回率，*F*1@*k* 是 Precision@*k* 和 Recall@*k* 的调和平均数。其表达式如下：

$$\text{Precision@}k = \frac{|\text{pred}_k(\text{vul}) \cap \text{label}(\text{vul})|}{|\text{pred}_k(\text{vul})|} \tag{14}$$

$$\text{Recall@}k = \frac{|\text{pred}_k(\text{vul}) \cap \text{label}(\text{vul})|}{|\text{label}(\text{vul})|} \tag{15}$$

$$F1@k = 2 \times \frac{\text{Precision@}k \times \text{Recall@}k}{\text{Precision@}k + \text{Recall@}k} \tag{16}$$

其中，$\text{pred}_k(\text{vul})$ 表示前 *k* 个预测结果，$\text{label}(\text{vul})$ 表示所有的真实漏洞库。

VLTA-LA 与另外 3 种漏洞识别方法的漏洞识别性能对比结果见表 8。在各项指标上 VLTA-LA 都达到了最优值，Precision@*k*、Recall@*k* 和 *F*1@*k* 的平均值分别达到了 0.75、0.87 和 0.78。相较于同为基于时间因子优化的轻量化第三方库漏洞识别模型 LightXML，VLTA-LA 的各项性能指标均提升了 10% 以上，且均优于基于树的分类器模型 FastXML 和 Bonsai。由此可见，使用时间因子优化推荐算法对候选迁移库进行筛选器后，可以去除大量无效迁移库，降低迁移语料库大小，从而有效增强模型的漏洞识别性能。

在与现有漏洞识别方法进行对比的基础上，本实验进一步验证了 VLTA-LA 方法在检测低空物联网中 3 类高危漏洞方面的效果，包括 GPS 欺骗（无人机）、时间敏感型漏洞以及内存泄漏，具体结果见表 9。

实验沿用表 8 中的 *k* 值设置，结果显示，VLTA-LA 方法在单一漏洞识别的平均精度（Precision@Avg）与平均召回率（Recall@Avg）均表现良好。尤其在地勘物联网中较为突出的 GPS 欺骗漏洞上，VLTA-LA 方法检测的平均精确度达到了 92%，体现出该方法在低空物联网场景下的有效性。此外，针对无历史记录的全新库等场景，实验从 DroneSec 数据集中选取了 20 个库模拟无历史记录情况，并借鉴文献[9]中的 CLIP 特征实现零样本迁移。实验结果表明，VLTA-LA 方法在此设定下的 *F*1 值达到了 71%。

此外，实验还比较了 VLTA-LA 与 LightXML 的执行时间，结果见表 10。在训练时间方面，LightXML 需要耗费 15 378.65 s，接近 4.3 h；而 VLTA-LA 仅须耗费 6 499.48 s，不到 2 h，时间缩短了约 58%。在预测时间与平均预测时间方面，VLTA-LA 分别需要耗费 0.49 ms 和 4.71 ms，而 LightXML 模型则分别需要耗费 103.72 ms 和 62.9 ms。相比于 LightXML，VLTA-LA 的时间分别缩短了 99.53% 和

表8　漏洞识别性能对比

指标	k	漏洞识别方法			
		FastXML	Bonsai	LightXML	VLTA-LA
Precision@k	1	0.81	0.87	0.88	0.98（+11.36%）
	2	0.59	0.62	0.64	0.73（+14.06%）
	3	0.45	0.47	0.49	0.53（+8.16%）
	平均值	0.62	0.65	0.67	**0.75（+11.94%）**
Recall@k	1	0.59	0.65	0.66	0.74（+12.12%）
	2	0.74	0.80	0.82	0.92（+12.19%）
	3	0.79	0.86	0.87	0.96（+10.34%）
	平均值	0.71	0.77	0.78	**0.87（+11.53%）**
$F1$@k	1	0.69	0.74	0.75	0.84（+12%）
	2	0.65	0/70	0.72	0.81（+12.5%）
	3	0.57	0.61	0.63	0.68（+7.93%）
	平均值	0.64	0.68	0.70	**0.78（+11.43%）**

表9　VLTA-LA对典型漏洞的检测效果

漏洞类型	Precision@Avg	Recall@Avg
GPS欺骗	0.92	0.88
协议过时	0.90	0.87
内存泄漏	0.86	0.85

表10　执行时间对比

模型	训练时间/s	预测时间/ms	平均预测时间/ms
LightXML	15 378.65	103.72	62.90
VLTA-LA	6 499.48（−57.7%）	0.49（−99.5%）	4.71

92.51%。以上结果表明，将时间因子优化推荐算法融入漏洞库识别模型，能够大幅降低模型的训练和对漏洞的识别时间。

进一步地，将本文算法模型与LightXML部署到真实低空物联网硬件平台STM32F4，从内存消耗、时延以及能耗等方面对比二者的性能，其结果见表11。实验结果表明，与LightXML相比，本文算法模型在资源效率方面具有显著优势：内存消耗从682 kbit降至328 kbit，降低了约51.9%；时延从60.31 ms降至8.53 ms，减少了约85.9%；能耗从44.3 mW降至9.7 mW，降低了约78.1%。这些数据充分证明，本文算法模型在资源受限的低空物联网环境中具有更好的适用性。

表11　物联网环境资源消耗对比

模型	内存消耗/kbit	时延/ms	能耗/mW
LighutXML	682	60.31	44.3
VLTA-LA	328	8.53	9.7

在实际应用中，本文所提方法可与现有安全检测工具协作，或作为现有工具的补充。具体实施方式包括：将模型检测出的高危结果推送至绿盟漏洞管理平台，对安全代码支持加密部署等。通过协同机制，能够提高漏洞检测效率，降低物联网环境的安全风险。本文所提方法可直接应用于低空物联网环境中的设备固件，对其代码进行漏洞检测，也可集成至开发环境中，在开发阶段为第三方库的选取选用提供推荐。

4　结束语

本文提出了一种基于时间因子优化推荐算法的迁移库漏洞识别方法。该方法通过构建时间因子优化推荐算法，筛选具有高时效性的迁移库，并采用结合知识蒸馏的轻量化Transformer模型实现漏洞识别。实验结果表明，VLTA-LA在推荐性能和漏洞识别效果上均优于现有对比方法，且显著降低了训练与识别过程中的时间开销，更适用于低空物联网环境。

未来的研究可以在以下几个方面开展：1) 在多终端轻量化适配方面，需要进一步探索更先进的模型压缩与量化技术，以适配更多类型的低空物联网终端芯片；2) 在动态时间窗口优化方面，当前时间窗口设置为固定值，未来可研究基于漏洞类型或库流行度的自适应时间衰减机制，以更精细地刻画风险演化过程；3) 在流程集成方面，可探索与DevSecOps流程的集成方案，即将VLTA-LA作为自动化安全组件集成至CI/CD流程中，为开发阶段提供实时的第三方库安全风险评估与迁移推荐。

参考文献：

[1] LARIOS VARGAS E, ANICHE M, TREUDE C, et al. Selecting third-party libraries: the practitioners' perspective[C]//Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM, 2020: 245-256.

[2] TEYTON C, FALLERI J R, BLANC X. Mining library migration graphs[C]//Proceedings of the 2012 19th Working Conference on Reverse Engineering. Piscataway: IEEE Press, 2012: 289-298.

[3] HE H, XU Y L, MA Y X, et al. A multi-metric ranking approach for library migration recommendations[C]//Proceedings of the 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). Piscataway: IEEE Press, 2021: 72-83.

[4] LI B, QUAN H W, WANG J W, et al. Neural library recommendation by embedding project-library knowledge graph[J]. IEEE Transactions on Software Engineering, 2024, 50(6): 1620-1638.

[5] JIANG T, WANG D Q, SUN L L, et al. LightXML: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2021, 35(9): 7987-7994.

[6] LYU Y B, LE-CONG T, KANG H J, et al. Chronos: time-aware zero-shot identification of libraries from vulnerability reports[C]//Proceedings of the 45th International Conference on Software Engineering. New York: ACM, 2023: 1033-1045.

[7] TANG B W, YAN L, ZHANG J, et al. Data-free generalized zero-shot learning[J]. arXiv preprint, 2024, arXiv: 2401.15657.

[8] ZHANG T, LIANG K M, DU R Y, et al. Disentangling before composing: learning invariant disentangled features for compositional zero-shot learning[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2025, 47(2): 1132-1147.

[9] CHEN Y, SANTOSA A E, SHARMA A, et al. Automated identification of libraries from vulnerability data[C]//Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice. New York: ACM, 2020: 90-99.

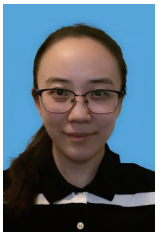
[10] ALRUBAYE H, MKAOUER M W, OUNI A. MigrationMiner: an automated detection tool of third-party Java library migration at the method level[C]//Proceedings of the 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME). Piscataway: IEEE Press, 2019: 414-417.

[11] PRABHU Y, VARMA M. FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning[C]//Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2014: 263-272.

[12] KHANDAGALE S, XIAO H, BABBAR R. Bonsai: diverse and shallow trees for extreme multi-label classification[J]. Machine Learning, 2020, 109(11): 2099-2119.

[13] JIANG T, WANG D Q, SUN L L, et al. LightXML: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2021, 35(9): 7987-7994.

[作者简介]



高燕（1981– ），女，中国电子科技集团公司第三十研究所高级工程师，主要研究方向为通信技术、计算机网络、网络安全。

罗琴（1981– ），女，博士，西南石油大学计算机与软件学院副研究员，主要研究方向为空间信息网络、网络安全。