基于图结构源代码切片的智能化漏洞检测系统

邹德清[1]，李响[2]，黄敏桓[2]，宋翔[3]，李浩[3]，李伟明[4]

（1. 华中科技大学网络空间安全学院，湖北 武汉 430074；
2. 信息系统安全技术重点实验室，北京 100101；
3. 华中科技大学软件学院，湖北 武汉 430074；
4. 华中科技大学网络与计算中心，湖北 武汉 430074）

摘 要：针对智能化漏洞检测，从源代码程序依赖图中根据漏洞特征提取图结构源代码切片，将图结构切片信息表征后利用图神经网络模型进行漏洞检测工作。实现了切片级的漏洞检测，并在代码行级预测漏洞行位置。为了验证系统的有效性，分别与静态漏洞检测系统、基于序列化文本信息和基于图结构化信息的漏洞检测系统做比较，实验结果表明，所提系统在漏洞检测能力上有较高准确性，并且在漏洞代码行预测工作上有较好表现。

关键词：漏洞检测；图结构；代码切片；深度学习

中图分类号：TP393

文献标识码：A

DOI: 10.11959/j.issn.2096−109x.2021088

Intelligent vulnerability detection system based on graph structured source code slice

ZOU Deqing[1], LI Xiang[2], HUANG Minhuan[2], SONG Xiang[3], LI Hao[3], LI Weiming[4]

1. School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
2.National Key Laboratory of Science and Technology on Information System Security, Beijing 100101, China
3.School of Software Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
4. Network and Computation Center, Huazhong University of Science and Technology, Wuhan 430074, China

Abstract: For the intelligent vulnerability detection, the system extracts the graph structured source code slices according to the vulnerability characteristics from the program dependency graph of source code, and then presents the graph structured slice information to carry out vulnerability detection by using the graph neural network model. Slice level vulnerability detection was realized and the vulnerability line was located at the code line level. In order to verify the effectiveness of the system, compared with the static vulnerability detection systems, the vulnerability detection system based on serialized text information, and the vulnerability detection system based on graph

收稿日期：2021−06−01；**修回日期：**2021−09−10

通信作者：李响，ideal_work@163.com

基金项目：国家自然科学基金（U1936211）

Foundation Item: The National Nature Science Foundation of China (U1936211)

论文引用格式：邹德清, 李响, 黄敏桓, 等. 基于图结构源代码切片的智能化漏洞检测系统[J]. 网络与信息安全学报, 2021, 7(5): 113-122.

ZOU D Q, LI X, HUANG M H, et al. Intelligent vulnerability detection system based on graph structured source code slice[J]. Chinese Journal of Network and Information Security, 2021, 7(5): 113-122.

structured information, the experimental results show that the proposed system has a high accuracy in the vulnerability detection capability and a good performance in the vulnerability code line prediction.

Keywords: vulnerability detection, graph structure, code slice, deep learning

1　引言

随着信息化进程不断加快，社会各个方面智能化程度越来越高，软件在生活、生产、学习等方面的应用越来越多，软件安全保障的需求量也随之增大，软件安全问题日益突显。其中，软件源代码漏洞检测的高效性和准确性急需提高以满足社会需求。例如，公民个人信息与财产大量通过网络传输与处理，为了保证公民信息与财产安全而带来的源代码漏洞检测的工作量十分巨大，而依赖于软件开发相关人员人工完成无法满足需求，此外，人工检测软件源代码漏洞受到多种因素影响，无法保证漏洞检测工作的准确性。所以，需要提高软件漏洞检测工作的智能化程度以保证其高效性和准确性。

目前，代码漏洞检测系统的研究主要包括传统漏洞检测方法和基于机器学习的漏洞检测方法。传统的漏洞检测方法主要有基于规则的漏洞检测方法和基于相似性的漏洞检测方法，基于规则的漏洞检测方法对于规则外漏洞存在较高漏报率，基于相似性的漏洞检测方法局限性较大。传统的将机器学习应用于代码漏洞检测的方法，漏洞数据的特征依赖于领域专家定义，需要花费大量人工投入，也正由于漏洞数据特征需要人工定义，对于特征的定义存在不可避免的缺失和误差，进而影响漏洞检测系统的准确率。基于深度学习的漏洞检测方法，由神经网络模型自动学习漏洞数据特征，可以避免大量人工投入，同时神经网络可以从更深层次分析漏洞数据特征，发现人类未定义的代码结构特征，从而有效降低漏洞检测系统的漏报率和误报率。

目前基于神经网络的漏洞检测方法的研究工作有以下两个方面。一方面，所使用的漏洞数据集主要是漏洞相关序列化代码或者经过一定处理的漏洞代码片段，提供给神经网络学习模型的漏洞数据集只有相关代码的文本信息，对于漏洞代码的依赖和调用信息没有很好地应用于预测源代码漏洞。模型得到的信息十分有限，并不能全面地获取漏洞特征，准确判断漏洞位置。另一方面，采用图结构表示源代码漏洞数据可以较好地表达漏洞特征信息[1]，已有相关研究将图结构化源代码漏洞数据应用到漏洞检测工作，但主要存在如下问题。

问题 1：采用的图结构化数据中包含较多与漏洞无关的源代码信息，这些信息作为神经网络模型的输入，最终影响漏洞检测系统的高效性和准确性。

问题 2：有关图神经网络模型应用于源代码漏洞检测领域的相关研究工作在漏洞检测粒度方面仍存在较大提升空间，不能精确定位漏洞位置，只能在较粗粒度上判断漏洞位置，对于漏洞的定位停留在文件级或者函数级。

本文主要构建了一个基于图结构化源代码切片的智能化漏洞检测系统。首先，系统在源代码漏洞数据集的预处理中采用代码切片技术，保证不会向神经网络模型输入过多与源代码漏洞无关的信息以提高系统的高效性和准确性；其次，系统可以在切片级别预测源代码漏洞并在代码行级定位漏洞位置，实现细粒度的源代码漏洞预测。

2　相关工作

2.1　传统的漏洞检测的相关工作

传统的漏洞检测方法主要有基于规则的漏洞检测方法和基于相似性的漏洞检测方法。基于规则的漏洞检测方法主要依赖定义的漏洞规则，漏洞规则的构建主要采用领域专家人工定义和编写程序从大量已知漏洞数据中总结两种方法[2]。首先，这种检测方法获取漏洞规则的过程需要投入大量人力，资源浪费严重；其次，这种检测方法的准确率基本取决于规则的完整性和正确性，而这种方法定义的漏洞规则的完整性和正确性很难得到保证。基于相似性的漏洞检测方法主要从代码复用的风险出发考虑可能触发的漏洞，局限性十分明显，对于不是代码复用引起的漏洞的漏报率很高[3]。例如，Xiao[4]等设计的 MVP 漏洞检测系统通过从漏洞样本数据中提取源代码漏洞的函

数特征及其修补函数的特征，将提取的特征与检测的目标函数匹配，进而在函数级别预测源代码漏洞，对于其已获取函数特征的漏洞有较好的检测效果，但是对于未从漏洞数据中提取到漏洞特征的相关漏洞没有实现有效检测。

2.2　基于机器学习的漏洞检测的相关工作

根据是否需要专家定义漏洞特征，基于机器学习的漏洞检测研究工作主要分为两种：一种是采用由专家定义漏洞特征的模型[5-8]；另一种是采用不依赖专家定义漏洞特征的深度学习模型[2,3,9-21]。一方面，采用由专家定义漏洞特征的模型的工作，先由专家定义漏洞特征，然后采用机器学习模型从输入数据中提取特征相关信息分析学习，最后达到检测源代码漏洞的目的。这种方法对于漏洞检测的效果主要依赖于专家定义的漏洞特征，具有较高的人工成本。此外，模型对于输入的漏洞相关数据，除了定义的漏洞特征相关信息外的其他漏洞信息没有得到充分利用，从而影响漏洞检测的准确性。另一方面，采用深度学习模型的漏洞检测工作，不需要由专家定义漏洞特征，既减少了人工参与，又可以使模型学习更深层次的漏洞特征，避免了人工定义特征带来的信息损失。根据采用的漏洞数据的结构不同，利用深度学习模型进行漏洞检测的工作主要有两类：一类是漏洞源代码数据采用序列化文本形式表示[2-3,9-18]；另一类是采用图结构化的漏洞源代码数据[19-21]。采用序列化文本形式表示漏洞源代码数据的相关研究工作主要是将漏洞源代码经过预处理，得到与漏洞相关的源代码切片、特征等信息，再将得到源代码文本信息经过表征后作为深度学习模型的输入。例如，陈肇炫[2]等设计的 Astor 系统基于源代码结构表征实现了切片粒度的漏洞检测工作；Li[3,10~11]等设计的 VulDeePecker 系统和μVulDeePecker 系统采用深度学习模型分别实现了函数调用类漏洞的检测工作和多种类型漏洞的检测工作，其设计的 SySeVR 系统和 VulDeeLocator 系统分别实现了切片级和代码行级的漏洞检测工作。这类研究在表征漏洞相关源代码信息时，对源代码结构信息有所缺失。为了克服模型对源代码结构信息学习不完整的问题，将漏洞源代码数据采用图结构表示应用于机器学习模型的训练可以更完整地表征源代码的结构信息。例如，Wang[19]等提出一个以门控图神经网络（GGNN，gated graph sequence neural networks）为基础的漏洞数据收集和漏洞检测系统，捕捉多种源代码漏洞关系，提高漏洞检测系统的准确性；Duan[20]等设计的 VulSniper 系统将源代码表示为代码属性图，将代码属性图表征为向量后输入注意力神经网络模型以实现判断源代码是否存在漏洞；Zhou[21]等设计的 Devign 系统提取多种图表示的源代码特征用于构造图结构化数据集并应用于门控图神经网络，创新性地增加了 Conv 层根据节点特征筛选节点，Devign 系统实现了在函数级别预测源代码漏洞。但以上研究工作在源代码漏洞预测的粒度局限于函数级，无法准确定位漏洞存在的代码行。本文设计的采用图结构化的代码切片数据训练神经网络模型，使模型可以学习到更完整的源代码结构信息，并且利用模型中的注意力机制定位漏洞代码行，提供更细粒度的漏洞检测信息。

3　智能化漏洞检测系统设计

本文的目标是设计一个漏洞检测系统，致力于源代码运行时的数据和控制依赖关系的表征和学习，并在低误报率和低漏报率的基础上实现漏洞智能检测并预测漏洞触发行。

本节介绍系统的设计与实现。图 1 是所提漏洞检测系统整体流程，首先获取漏洞候选图结构化源代码切片的过程，之后训练神经网络模型的工作流程。

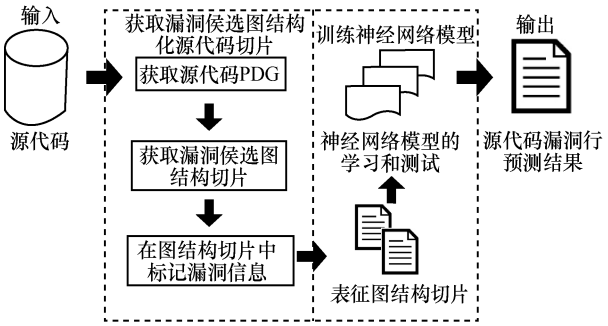


图 1　漏洞检测系统整体流程

Figure 1　Vulnerability detection Process of the system

3.1　漏洞候选源代码图结构化切片的获取

首先获取源代码对应的程序依赖图（PDG，

program dependence graph)，之后从源代码 PDG 中获取漏洞候选源代码图结构化切片，最后，在图结构切片中标记漏洞信息。

（1）获取源代码 PDG

检测系统的输入为漏洞检测目标源代码，分析程序源代码的控制流、数据流等信息，得到以源代码中函数为单位的 PDG。对函数级 PDG 进行信息的完善，补充函数中调用其他函数的相关信息，使函数运行时发生调用函数动作的相关信息可以追踪，实现跨函数的 PDG。获得 PDG 中一个节点对应的是源程序中一行代码（节点中包含关于本节点源代码语句的相关信息：语句类型，节点 ID，在源程序中的位置，所属函数）；PDG 中还包含函数运行时的控制流和数据流关系以及节点之间边的关系表示。

（2）获取漏洞候选图结构切片

根据 4 种漏洞语法特征：（库/API 函数调用、数组使用、指针使用和表达式定义），从上一步获得的 PDG 中确定候选漏洞节点，即节点对应的代码语句中如果符合漏洞语法特征，则确定为候选漏洞节点。根据候选漏洞节点从 PDG 中截取与候选漏洞节点存在数据依赖或者控制依赖关系的节点和相应的边构成漏洞候选图结构化切片。例如，某个节点对应的源代码语句中存在指针使用，这个节点就会被确定为漏洞候选节点，从这个节点出发，寻找与其存在控制或者数据依赖关系的节点，从上一步得到的 PDG 中提取出与本节点相关的节点集，同时提取出节点集所包含的边组成一个 PDG 子图，这个子图本质上就是源程序的一个切片，子图中所包含的代码行集合就是源程序中代码行的一个子集，这个 PDG 子图就是漏洞候选源代码切片之一。

（3）在图结构切片中标记漏洞信息

为了训练模型，系统的输入源程序包含有漏洞的源程序及其修补程序和无漏洞的源程序，源程序已经标注是否有漏洞，有漏洞源程序已经标注漏洞触发代码行。上一步得到的漏洞候选源代码切片包含的节点中，如果存在节点对应的代码行有标注的漏洞触发代码行，则这个节点标注为漏洞触发节点，节点所属的切片标注为存在漏洞的代码切片。

3.2　神经网络模型

图 2 为模型训练和测试阶段，输入分别为训练集数据或者测试集数据，首先表征漏洞候选图结构切片，将图信息表征为可以作为神经网络模型学习的数据，之后将表征后的信息输入神经网络模型，输出完成训练的神经网络模型；测试部分则输出测试集数据的漏洞检测及漏洞行预测结果。

3.2.1　漏洞候选图结构切片表征信息

（1）映射代码中的自定义符号

训练集和测试集中输入的图结构切片中，包含的代码文本信息经过表征后才能输入神经网络模型进行测试和训练。这些代码文本信息需要表征为向量，这个过程采用相关的神经网络模型完成。而输入数据集的代码中包含大量软件工程师自定义的符号，如自定义的变量名、函数名等。这些人工自定义的符号与漏洞无关，在表征时会被视为唯一的值，对于将代码文本表征为向量的神经网络模型会影响其准确性。所以，自定义的符号将会被映射为统一的符号代替。例如，变量将被映射为 VAR1，

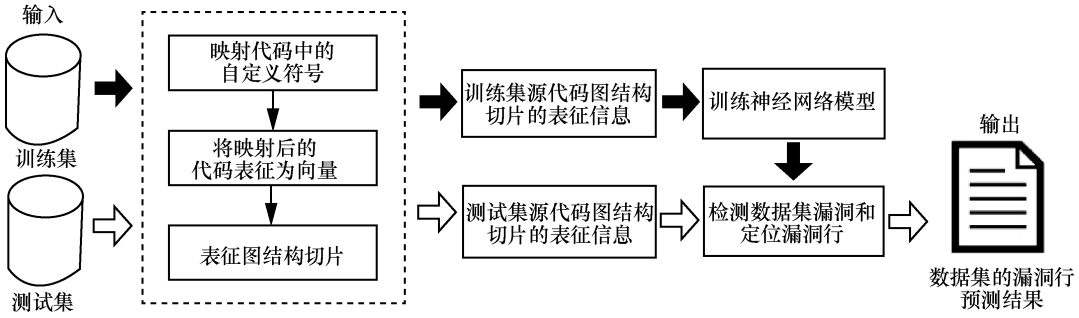


图 2　模型训练和测试阶段

Figure 2　Model training and testphase

VAR2…… 函数名将被映射为 FUN1，FUN2……。在不同的切片中出现的变量、函数等可能会被映射为相同的符号代替，以此避免无限制的自定义变量对将代码文本映射为向量准确性的影响。

（2）将映射后的代码表征为向量

首先，将代码文本中的具体代码行分为一个符号集。例如，“int VAR1，VAR2；”会被分为“int”“VAR1”“，”“VAR2”“；”。之后，为了保证每行代码转换后的向量长度一致，将对一行代码所对应的符号集进行补全或者截断操作，使一行代码对应的符号集所包含的符号数量保持一致，从而保证每一行代码转换为向量后的长度一致。本文采用 Word2Vec 工具将符号集转换为向量，Word2Vec 在经过训练后，可以将符号一一对应为相应的向量，满足本文系统的需要。

（3）表征图结构切片

输入神经网络模型的图信息主要包括切片的图结构信息和节点中的信息。节点对应的代码行内容已经表征为向量。切片的图结构信息包括：节点的唯一标识信息、节点所属的图标识信息、节点间关系的标识信息、图的分类标识信息。节点中的信息包括：节点对应的代码行内容和节点对应的代码行在源程序中的位置，

输入模型的数据包含多个图（即图结构化切片）和多个图对应的许多节点，这些节点的信息将一同输入模型，因此需要对节点唯一标识，并标识节点所属的图，即当前节点属于哪一个图。节点的唯一标识信息将映射为节点输入模型的排序，如节点 a 的相关描述信息在输入时排序为 1，则表示节点 a 的唯一标识为 1。

节点所属的图标识信息将采用图的唯一标识对节点一一标识，如节点的图标识信息文件中第 1 行的信息为“3”，则节点标识为 1 的节点属于图标识为 3 的图。

输入模型的所有图中节点之间的关系也是统一标识。节点之间关系的标识采用邻接矩阵的稀疏矩阵的简单表示形式，即采用边对应的两个节点的唯一标识来表示两个节点之间的关系。例如，节点标识对[1，2]表示：节点标识 1 和节点标识 2 所属的图中存在一条由节点 1 指向节点 2 的边，节点所属的图即表明边所属的图。

输入模型的所有图中节点对应的代码行将一起输入模型。节点对应的代码行内容被表征为向量，代码行向量输入的顺序和节点的图标识顺序一致，节点 1 对应的代码行向量输入时排序为 1。

神经网络模型每次输入包含多个图的样本，每次输入的图结构切片统一编码，每一个图结构切片有唯一的键值，图的分类标识输入的顺序即代表图的唯一标识键值。其中，标识为“2”代表对应的图结构切片中存在漏洞节点，标识为“1”代表对应的图结构切片中没有漏洞节点。例如，输入顺序为 1 的图结构切片分类标识为“2”，表示图唯一标识为 1 的图结构切片存在漏洞节点。

节点对应的代码行在源程序的位置信息的排序顺序和节点的唯一标识一致，可以根据节点的标识信息映射到节点对应的代码行在源程序中的位置。

总体来说，节点和图的相关信息输入顺序即表示其唯一标识，如节点 a 的某个信息输入时的排序为 1，则节点 a 所有相关信息在输入的排序均为 1，图亦是如此。

3.2.2　神经网络模型训练

图 3 为本文系统采用的神经网络模型架构，将图结构切片表征后作为神经网络模型的输入，图结构切片中的每一个节点可以映射到源代码中的一行代码。

模型采用全局池化和分层池化结合的思想，模型全局由 3 个“图卷积–池化”模块组成，每个模块中包含图卷积层和池化层。每个模块池化之后的节点特征经过 ReadOut 层聚合最终在线性层（MLP）连接进行图分类操作。本文系统采用 3 个“图卷积–池化”模块组成的主要原因是，3 个模块的漏洞检测效果好于一个模块或者两个模块组成的神经网络模型，在 3 个模块的基础上继续增加组成模块数量在漏洞检测效果上的提升并不明显，但是效率会随着模块数量的增加而逐渐降低。

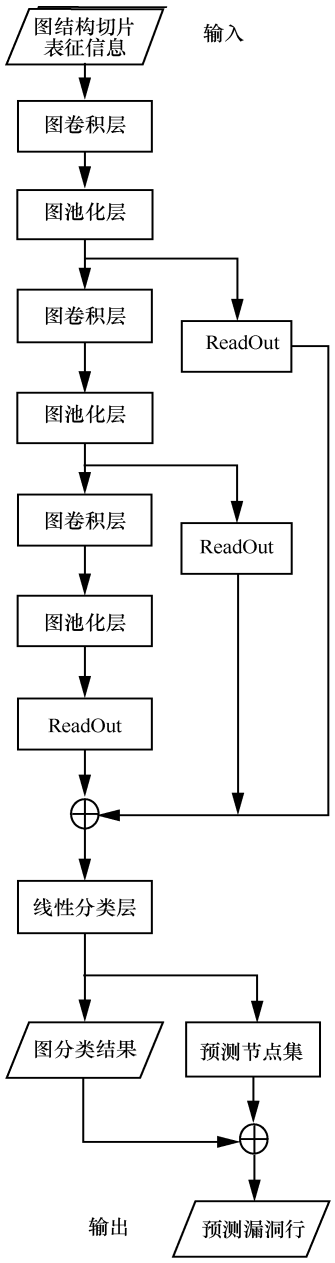


图 3　神经网络模型架构

Figure 3　Neural network model architecture

图卷积层：采用自注意力机制（self- attention），本模型中的自注意力机制是一种全局图注意力机制（global graph attention），全局图注意力机制即考虑全图所有节点的影响，计算相邻节点之间的影响，经过多次计算后，使每个节点获取图的全局几何结构。图卷积层由多次卷积操作组成，每一次卷积操作每一个节点都从图中与自己相邻的节点获取其信息并据此更新自身信息，经过多次卷积操作后，每一个节点可以有效感知整个图中的所有关系和节点信息并完成自身节点信息的更新。

池化层：根据上一次卷积层计算后更新的各个节点的信息筛选出对于最终漏洞预测影响较大的节点，筛选出的节点组成的子图将作为下一次循环的输入，本次池化层计算筛选的结果会经过 ReadOut 层读出，并作为最终图分类计算的考虑因素之一。

经过多次卷积、池化和 ReadOut 循环后，筛选后的节点越来越少，对于最终漏洞预测的影响越来越大，但是之前被筛选的节点信息并不会丢弃，而是通过每次 ReadOut 保存下来，经过一定权重计算作为图分类的依据之一。同样，经过多次筛选后的节点会更大可能预测为漏洞节点。以上每次 ReadOut 之后的结果输入最终的多层感知机计算后做出图分类结果。

ReadOut 之后筛选出的节点将结合图分类结果，则筛选出的节点对应到属于该图的节点，并对应到节点对应的代码行，即漏洞代码行。

3.3　神经网络模型测试

模型的测试阶段将一组源代码数据经过分析后，生成相应的 PDG，根据 4 类关注点提取出图结构切片，然后映射切片中的代码，之后表征图结构切片，得到的数据作为完成训练模型的输入，分析模型对于输入数据的判断结果。模型输出的结果分为两类：一类为模型判断为没有源代码漏洞的图结构切片；另一类为模型判断为存在源代码漏洞的图结构切片，这部分数据的输出还会包含模型预测存在较大可能导致源代码漏洞的节点，根据节点信息可以映射为源代码中具体的代码行，分析测试结果判断系统对于源代码漏洞和具体漏洞易发代码行预测的准确性。

模型分析节点对于漏洞预测结果的重要性，并充分考虑图中全局节点对于预测代码漏洞的影响。模型根据包含丰富代码结构信息的数据学习形成较高准确率的漏洞判断能力。

4　实验设计与结果分析

本文所设计的实验用于回答以下 3 个研究问题。

RQ1：本文系统相较于采用序列化结构源代

码切片信息的深度学习漏洞检测系统，本文系统采用图结构化源代码切片信息是否有更好的漏洞检测性能？

RQ2：相较于其他基于图神经网络模型的漏洞检测系统是否有更好的漏洞检测性能？

RQ3：本文系统相较于静态漏洞检测系统，是否有更好的漏洞检测性能？

4.1　评价漏洞检测系统准确性的标准

本文主要采用以下指标来评价漏洞检测系统。

精度：反映系统判断为漏洞的样本中正确预测漏洞的比例。

$$P=\frac{\mathrm{TP}}{\mathrm{TP+FP}}$$

召回率：反映出检测系统判定存在漏洞样本占实际存在漏洞样本的比例。

$$R=\frac{\mathrm{TP}}{\mathrm{TP+FN}}$$

$F1$ 值：综合考量了准确率和召回率，更加均衡地反映出系统的漏洞检测性能。

$$F1=\frac{2\cdot P\cdot R}{P+R}$$

IoU 值：反映系统检测存在漏洞样本中漏洞行位置的准确性。本文中的 IoU 值为计算每个样本的 IoU 值，求和取平均之后的值。

$$\mathrm{IoU}=\frac{|U\bigcap V|}{|U\bigcup V|}$$

$|V|$值：系统预测样本中漏洞行数量的平均值。

$$|V|=\frac{M}{N}$$

其中，TP 为系统判断为存在漏洞且确实存在漏洞的样本数量；FP 为系统判断为存在漏洞而没有漏洞的样本数量；FN 为系统判断没有漏洞而真实存在漏洞的样本数量；U 为存在漏洞样本中真实的漏洞行；V 为系统定位出漏洞样本中的漏洞行；M 为系统判断样本中漏洞行的总数；N 为样本总数。

4.2　图结构化数据集的生成

本文实验采用的源代码数据集来源于 SySeVR[11]中所使用的 SARD（software assurance reference dataset）。SARD 是由人工编写的漏洞代码得到的测试用例，数据集包含有漏洞的源程序、无漏洞的源程序、修补过漏洞的源程序，并且数据集中的漏洞源程序已经标识漏洞代码行。将数据集处理为可以输入本文系统神经网络模型的数据格式的过程如下。

（1）从源程序数据获取漏洞候选图结构切片

首先基于 Joern 工具将源程序解析为以函数为单位的 PDG 图，然后根据 4 种漏洞语法特征：（库/API 函数调用、数组使用、指针使用和表达式定义）确定 PDG 图中的候选漏洞节点，再根据候选漏洞节点选出与漏洞节点存在控制依赖和数据依赖的节点，最后从 PDG 图中获取由这些节点组成的图结构切片，图结构切片中包含图的结构信息、节点对应的代码行内容、节点对应代码行在源程序中的位置、节点对应代码行的语句类型、图结构切片是否包含漏洞的标记、图中包含的漏洞节点。获取源程序 PDG 图的过程是将 Joern 解析后的数据由 Neo4j 图形数据库读出，然后采用 Python 编写的程序通过 Neo4j 数据库读取数据，完成提取漏洞候选图结构切片。

（2）表征图结构切片信息

本文系统采用 Python 编写的程序实现设计部分提出的映射代码语句的机制，之后构建 Word2Vec 模型将映射后的代码符号集映射为可以输入神经网络模型的向量格式，将代码符号映射为向量包括如下内容。①构建语料库：对实验数据中的漏洞候选图结构切片中的代码行做分词处理，将代码行分成符号集，将所有的符号构建成预料库存。②训练 Word2Vec 模型：将语料库输入 Word2Vec 模型训练，本文设置了 5 轮迭代训练得到最终的模型。③将所有漏洞候选图结构切片中节点所包含代码行通过完成训练的 Word2Vec 模型表征为向量。

输入神经网络模型的数据包含多个漏洞候选图结构切片，将其节点信息、节点关系信息、图分类信息、节点−图对应关系信息经过表征后分别存储为一个文件，作为神经网络模型的输入。采用设计部分描述的方法表征图结构信息，以上过程采用 Python 编写的程序实现。

4.3　实验与结果分析

实验采用的数据中，SARD 共生成 122 103 个图结构化切片。实验数据中的 80%用于训练神经网

络模型，10%用于验证模型使用，其余的 10%用于测试模型的各项性能数据。实验环境的主要配置参数如下。硬件配置：CPU 为 Intel Xeon E5-2620 2.40 GHz，GPU 为 NVIDIA GeForce GTX 1080，内存 32 GB，磁盘大小 2 TB；操作系统为 Linux 4.18.0-269.el8.x86_64。实验主要对提出的 3 个研究问题进行针对性实验。

（1）针对 RQ1 的实验

为了确定采用图结构表示源代码切片信息，并基于图神经网络模型构建漏洞检测系统能否更好地检测源代码漏洞，设计了本文系统和同样采用源代码切片作为神经网络模型输入样本的漏洞检测系统 SySeVR[11]作比较。SySeVR 的不同点在于其采用序列化结构表示源代码切片，将源代码切片的代码内容作为文本信息表征后输入神经网络模型。

本文系统采用图神经网络模型中的 3 个“图卷积–池化”模块，每个模块都会对图中的节点按照固定比例进行筛选，这个比例就是池化比率（pooling ratio）。为了确定最优结果的池化筛选比例，设计了针对性实验，比较在不同池化筛选比例下模型各项参数的结果，最终选择最佳比例。

根据表 1 给出的实验结果，综合考虑所有评价标准，选择在池化比率为 0.65 条件下为本文系统最优结果，后面的实验将采用在这一条件下的实验结果与其他漏洞检测系统进行对比。

表 1　本文系统在不同池化比率下各指标对比结果
Table 1　Index comparison results of the proposed system under different ratios

池化比率	*P*	*R*	*F*1	IoU	|*V*|
0.50	91.7%	85.4%	88.4%	12.3%	2.2
0.60	93.2%	84.8%	88.8%	13.1%	3.7
0.65	92.8%	86.8%	89.7%	13.4%	4.7
0.70	89.2%	89.4%	89.3%	12.5%	5.5
0.80	93.7%	85.4%	89.4%	11.3%	8.1

表 2 给出了针对 RQ1 的实验结果。①本文系统的 *F*1 值相较于 SySeVR 系统高 0.4%，说明本文系统的判断样本漏洞的综合能力相较于 SySeVR 有略微提升。本文系统的精度相比 SySeVR 提高 8.2%，说明本文系统判断样本是否存在漏洞的精确度有明显提升，主要原因在于本文系统采用图结构表示代码切片，可以为神经网络模型提高更丰富的数据依赖和控制依赖关系，模型可以据此学习到多方面的漏洞特征。SySeVR 采用文本形式表征代码切片，提供给模型学习的信息十分有限，模型只能通过代码文本信息总结漏洞特征，相对于图神经网络模型总结的漏洞特征有所缺失，所以本文系统在判断样本是否存在漏洞时会考虑更多因素，做出更加准确的判断，有更高的精度，但同时存在一些漏洞的样本无法被准确判断出漏洞，导致召回率有所降低。②本文系统的 IoU 值相较于 SySeVR 系统提高 5.5%，|*V*|则由 13.3 降至 4.7，可以在更精确位置定位漏洞，提供更加有效的漏洞预测信息。本文系统可以在代码行粒度预测漏洞位置，而 SySeVR 只能在切片级粒度预测漏洞，且本文漏洞检测精度较高，所以本文系统在定位漏洞位置方面相对于 SySeVR 有较大提升。

表 2　本文系统和 SySeVR 系统的各指标对比结果
Table 2　Index comparison results of between the proposed system and SySeVR system

检测系统	*P*	*R*	*F*1	IoU	|*V*|
SySeVR 系统	84.6%	94.5%	89.3%	7.9%	13.3%
本文系统	92.8%	86.8%	89.7%	13.4%	4.7%

（2）针对 RQ2 的实验

为了确定本文系统相较于其他基于图神经网络模型的漏洞检测系统是否有更好的漏洞检测性能，设计了本文系统和 Devign 系统比较。

表 3 给出了针对 RQ2 的实验结果。本文系统相较于 Devign 系统，精度和 *F*1 值分别提高 56.7%和 36.6%，主要原因是本文在数据预处理阶段，会根据 4 类漏洞特征提取源代码切片，而 Devign 系统中输入图神经网络模型的数据是以函数单位的，对比之下，本文系统在输入模型的数据中一定程度上减少了与漏洞无关的源代码信息的输入，提高了系统漏洞检测的准确性。本文系统利用神经模型自注意力机制在池化阶段对图中节点进行筛选，间接对一个样本中的代码行进行筛选，使模型可以更加专注于学习与漏洞有关的代码行特征，从而更加精确地判断样本是否包含漏洞。Devign 系统是对函数整体进行学习，一个函数中不一定所有的代码行都与漏洞相关，相对容易学习到与漏洞无关代码特征，这会导致将无漏洞样本判断为存在漏洞样本。

表 3　本文系统和 Devign 各指标对比结果
Table 3　ndex comparison results of between the proposed system and Devign system

检测系统	*P*	*R*	*F*1
Devign 系统	36.1%	100.0%	53.1%
本文系统	92.8%	86.8%	89.7%

（3）针对 RQ3 的实验

将本文系统分别和以下几种漏洞检测系统做比较。①Checkmarx，一款商业的代码静态分析工具。②Flawfinder，一款开源的关于 C/C++ 静态扫描分析工具。③RATS，一款开源的代码静态分析工具。以上工具均可应用于本文系统训练和测试所使用的数据集，且在源代码静态扫描领域具有较好表现。

表 4 给出了针对 RQ3 的实验结果。根据实验结果可知，本文系统相较于对比实验中结果较好的静态代码分析工具 Checkmarx，精度和召回率分别提高 61.9%和 43.6%，这是因为静态漏洞检测系统主要是基于规则和重复性的漏洞检测器，由于定义规则的不完整性，系统会产生较高的假阳率和假阴率。而采用图神经网络模型自动学习漏洞特征可以避免由人类专家定义漏洞特征带来的不完整性，从而总结出更深层次的漏洞特征，提高漏洞检测系统的准确性。本文系统的 *F*1 值提升至 89.7%，相对于静态检测工具漏洞检测综合能力提高两倍以上，证明将图神经网络模型应用于漏洞检测检测领域是可行的，并且具有较高潜力。

表 4　4 种漏洞检测系统的各指标对比结果
Table 4　Index comparison results of four vulnerability detection system

检测系统	*P*	*R*	*F*1
RATS 系统	12.8%	14.7%	13.7%
Flawfinder 系统	22.8%	29.6%	25.7%
Checkmarx 系统	30.9%	43.2%	36.1%
本文系统	92.8%	86.8%	89.7%

5　结束语

本文提出了基于图结构源代码切片的智能化漏洞检测系统，主要将从源代码中获取的图结构切片的信息经过表征后，由图神经网络模型学习漏洞相关特征，最终模型可以实现预测源代码切片是否存在漏洞和预测含有漏洞的源代码切片中的漏洞行。经过和其他漏洞检测工具的性能对比实验，证实将图神经网络模型应用于源代码漏洞检测工作是一种有效的方法，且是一个十分有潜力的研究方向。

本文提出的漏洞检测系统在定位漏洞触发行时采用模型多次按比例进行节点筛选，最终筛选得到的节点对应到代码行即为预测的漏洞行。这种方法对于代码行数量较大的检测样本，预测出的漏洞行也会随之变多，未来可以针对这一问题开展研究，使漏洞行定位预测变得更加准确。

参考文献：

[1] 李韵, 黄辰林, 王中锋, 等. 基于机器学习的软件漏洞挖掘方法综述[J]. 软件学报, 2020, 31(7): 2040-2061.
LIY, HUANG C L, WANG Z F, et al. Survey of software vulnerability mining methods based on machine learning[J]. Journal of Software. 2020, 31(7): 2040-2061.

[2] 陈肇炫, 邹德清, 李珍, 等. 基于抽象语法树的智能化漏洞检测系统[J]. 信息安全学报, 2020, 5(4): 1-13.
CHEN Z X, ZOU D Q, LI Z, et al. Intelligent vulnerability detection system based on abstract syntax tree[J]. Journal of Cyber Security. 2020, 5(4): 1-13.

[3] LI Z, ZOU D Q, XU S H, et al. VulDeePecker: a deep learning-based system for vulnerability detection[C]//The Network and Distributed System Security Symposium. 2018.

[4] XIAO Y, CHEN B H, YU C D, et al. MVP: Detecting vulnerabilities using patch-enhanced vulnerability signatures[C]//USENIX Security Symposium. 2018.

[5] GUSTAVO G, GUILLERMO L G, UZAL L, et al. Toward large-scale vulnerability discovery using machine learning[C]//The 6th ACM on Conference on Data and Application Security and Privacy (CODASPY'16). 2016: 85-96.

[6] STEPHAN N, THOMAS Z, CHRISTIAN H, et al. Predicting vulnerable software components[C]//2007 ACM Conference on Computer and Communications Security (CCS'07). 2007: 529-540.

[7] FABIAN Y, MARKUS L, KONRAD R. Generalized vulnerability extrapolation using abstract syntax trees[C]//The 28th Annual Computer Security Applications Conference. 2012: 359-368.

[8] FABIAN Y, ALWIN M, HUGO G, et al. Automatic inference of search patterns for taint-style vulnerabilities[C]//IEEE Symposium on Security and Privacy (S&P'15). 2015: 797-812.

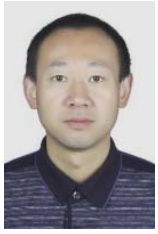
[9] ZOU D Q, WANG S J, XU S H, et al. μVulDeePecker: a deep learning-based system for multiclass vulnerability detection[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 15(5): 2224-2236.

[10] LI Z, ZOU D Q, XU S H, et al. VulDeeLocator: a deep learning-based fine-grained vulnerability detector[J]. IEEE Transactions on Dependable and Secure Computing, 2021.

[11] LI Z, ZOU D Q, XU S H, et al. SySeVR: a Framework for using deep learning to detect software vulnerabilities[J]. IEEE Transac-

tions on Dependable and Secure Computing, 2021.
[12] LIN G J, XIAO W, ZHANG J, et al. Deep Learning-based vulnerable function detection: a benchmark[C]//The 21st International Conference on Information and Communications Security. 2019: 219-232.
[13] LIN G J, ZHANG J, LUO W, et al. POSTER: vulnerability discovery with function representation learning from unlabeled projects[C]// ACM SIGSAC Conference on Computer and Communications Security. 2017: 2539-2541.
[14] LIN G J, ZHANG J, LUO W, et al. Cross-project transfer representation learning for vulnerable function discovery[J]. IEEE Trans Industrial Informatics, 2018, 14(7): 329-3297.
[15] LIU S G, LIN G J, QU L Z, et al. CD-VulD: cross-domain vulnerability discovery based on deep domain adaptation[J]. IEEE Transactions on Dependable and Secure Computing, 2020.
[16] VAN N, TRUNG L, TUE L, et al. Deep domain adaptation for vulnerable code function identification[C]//International Joint Conference on Neural Networks. 2019: 1-8.
[17] REBECCA R, LOUISK, LEI H, ET A L. Automated vulnerability detection in source code using deep representation learning[C]//The 17th IEEE International Conference on Machine Learning and Applications(ICMLA). 2018: 757-762.
[18] TIM S. Machine-learning supported vulnerability detection in source code[C]//The 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering(ESEC/FSE). 2019: 1180-1183.
[19] WANG H T, YE G X, TANG Z Y, et al. Combining graph-based learning with automated data collection for code vulnerability detection[J]. IEEE Transactions on Dependable and Secure Computing. 2020.
[20] DUAN X, WU H, JI S L, et al. VulSniper: focus your attention to shoot fine-grained vulnerabilities[C]//The 28th International Joint Conference on Artificial Intelligence. 2019: 4665-4671.
[21] ZHOU Y Q, LIU S Q, SIOW J, et al. Devign: effective vulnerability identification by learning comprehensive program semantics via graph neural networks[C]//Annual Conference on Neural Information Processing Systems(NeurIPS). 2019: 10197-10207.

[作者简介]



邹德清（1975− ），男，湖南浏阳人，华中科技大学教授、博士生导师，主要研究方向为系统与软件安全、云计算安全。

李响（1983− ），女，河北涿州人，信息系统安全技术重点实验室高级工程师，主要研究方向为软件脆弱性分析、恶意代码检测。

黄敏桓（1971− ），男，江西万载人，信息系统安全技术重点实验室研究员，主要研究方向为脆弱性自动分析、网络安全评估与验证。

宋翔（1996− ），男，河南新乡人，华中科技大学硕士生，主要研究方向为网络安全。

李浩（1995− ），男，河南驻马店人，华中科技大学硕士生，主要研究方向为网络安全。

李伟明（1975− ），男，湖南株洲人，华中科技大学副教授，主要研究方向为网络安全。