计算机科学
COMPUTER SCIENCE

基于 QRNN 的网络协议模糊测试用例过滤方法

胡志濠，潘祖烈

引用本文

胡志濠，潘祖烈. 基于 QRNN 的网络协议模糊测试用例过滤方法[J]. 计算机科学, 2022, 49(5): 318-324.

HU Zhi-hao, PAN Zu-lie. Testcase Filtering Method Based on QRNN for Network Protocol Fuzzing[J]. Computer Science, 2022, 49(5): 318-324.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210300281

基于 QRNN 的网络协议模糊测试用例过滤方法

胡志濠　潘祖烈
国防科技大学电子对抗学院　合肥 230037
安徽省网络空间安全态势感知与评估重点实验室　合肥 230037
(huzhihao@nudt.edu.cn)

摘　要　目前,网络协议模糊测试的目标趋向于大型协议实体,而传统的测试用例过滤方法主要是基于测试对象的运行状态信息,测试对象越庞大,其执行单个测试用例的时间也越长。因此,针对传统的网络协议模糊测试用例过滤方法存在无效执行时间长、效率低下的问题,依据循环神经网络模型对序列数据较强的处理和预测能力,提出一种基于 QRNN 的网络协议模糊测试用例过滤方法。通过学习网络协议的结构特征,包括字段取值范围和字段间约束关系,该方法可以自动过滤无效测试用例,减少协议实体测试用例的执行次数。实验结果表明,与传统的网络协议模糊测试用例过滤方法相比,所提方法可以有效降低网络协议漏洞挖掘的时间成本,显著提高网络协议模糊测试的效率。

关键词:测试用例过滤;QRNN;网络协议;模糊测试;深度学习
中图法分类号　TP393

Testcase Filtering Method Based on QRNN for Network Protocol Fuzzing

HU Zhi-hao and PAN Zu-lie
College of Electronic Engineering,National University of Defense Technology,Hefei 230037,China
Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation,Hefei 230037,China

Abstract　At present,targets of network protocol fuzzing tend to be large protocol entities,and traditional testcase filtering methods are mainly based on the running status information of the test object. The larger the test object,the longer it takes to execute a single testcase. Therefore,in view of the problems of long invalid execution time and low efficiency in traditional testcase filtering methods for network protocol fuzzing,a testcase filtering method based on QRNN for network protocol fuzzing is proposed according to strong abilities of recurrent neural network models to process and predict sequence data. The method can automatically filter invalid testcases by learning structural characteristics of the network protocol,including the value range of fields and constraint relationships between fields,and reduce the number of testcases executed by the protocol entity. Experimental results show that,compared with traditional testcase filtering methods for network protocol fuzzing,the proposed method can effectively reduce the time cost of network protocol vulnerability discovery and dramatically improve the efficiency of network protocol fuzzing.

Keywords　Testcase filtering,QRNN,Network protocol,Fuzzing test,Deep learning

1　引言

网络协议作为网络的基础,其不正确实现触发的安全漏洞通常会导致严重的后果[1]。2017 年,不法分子通过改造名为“永恒之蓝”的网络攻击工具制作了 WannaCry 勒索病毒,该病毒利用 Windows 系统的 SMB 协议漏洞进行主动传播和自我复制。短短几天内,70 多个国家和地区的 10 万余台电脑遭受攻击,大量重要文件被加密,给全球数以亿计的计算机用户造成了巨大的经济损失。因此,网络协议的安全性测试成为网络与信息安全领域的重大课题和研究热点。

随着软件规模的不断扩大和复杂度的提高,模糊测试在网络协议的安全性测试方面表现出无可比拟的优势,是当今针对网络协议最常用也最有效的漏洞挖掘方法。网络协议模糊测试的主要原理是通过生成或变异的方式构造测试用例,然后通过套接字向协议实体发送测试用例,监视协议实体以发现网络协议在实现时存在的漏洞。测试过程中的关键是遵守协议实体的协议规范和状态机制,以产生能被协议实体接收处理的测试用例[2]。

根据测试用例生成策略,网络协议模糊测试可以分为基于生成的网络协议模糊测试和基于变异的网络协议模糊测试两类[3],其中基于生成的网络协议模糊测试又可进一步细分为半自动化和全自动化两类。以 Peach[4] 为代表的基于生成

到稿日期:2021-03-29　返修日期:2021-07-14
基金项目:国家重点研发项目(2017YFB0802900)
This work was supported by the National Key R&D Program of China(2017YFB0802900).
通信作者:潘祖烈(panzulie17@nudt.edu.cn)

的网络协议半自动化模糊测试方法虽然在一定程度上可以提高测试用例的有效性，但前期往往需要耗费大量人力与时间成本，以研究测试对象的协议规范，导致其实用价值受到影响；而以 beSTORM[5] 为代表的基于生成的网络协议全自动化模糊测试方法，尽管能以自动化的方式将协议规范转换成测试用例集，但因其由企业研发，通常不公开源代码，导致无法被二次开发，可扩展性低。相比之下，以 AFL_{NET}[6] 为代表的基于变异的网络协议模糊测试方法无须人工参与确定输入规约以及生成测试用例等，自动化程度高，并开放源代码，可轻松扩展以支持指定功能，因此成为目前使用最为广泛的网络协议模糊测试方法。但另一方面，基于变异的网络协议模糊测试方法往往盲目地修改测试用例的数据，虽能保证测试用例的畸形度，但因变异策略过于随机，仅有少量的测试用例符合协议实体的接收要求[7]。而有效测试用例同时具有良好的接收率和畸形度，由此可见，协议实体需要执行大量的无效测试用例，这是一个低效的过程。因此，如何优化测试用例过滤方法，保证执行最小测试用例集合就能覆盖最多的路径或触发最多的潜在漏洞，是网络协议模糊测试研究的热点之一。

近年来，国内外学者开始研究如何将机器学习技术应用于测试用例过滤。2017 年，Gong 等基于模糊测试工具 AFL 生成的测试用例训练出了一个深度学习模型[8]，该模型能够预测 AFL 新一轮生成的测试用例能否改变程序状态。因此，AFL 只需要执行能够产生新状态的测试用例，自身的效率就能得到提高。2018 年，Karamcheti 等提出一种基于机器学习对程序行为直接建模的灰盒模糊测试方法[9]。学习得到的前向预测模型将程序输入映射到执行轨迹，执行轨迹分布的熵可用来评估模型对输入的确定性高低，熵越大（不确定性越高），表明输入在执行过程中有可能覆盖新的代码区域。该方法通过关注不确定的输入，忽略确定的输入，显著减少了不必要的执行次数，提高模糊测试的效率。2020 年，Zong 等提出一种基于深度学习的方法 FuzzGuard[10]。该方法无须执行目标程序就能预测输入的可达性，从而帮助导向式灰盒模糊测试工具 AFLGo 过滤掉不可达的输入，效率显著提高。总体来看，以上研究成果都是以普通二进制程序为目标，而针对网络协议模糊测试用例过滤的研究仍处于起步阶段。

为弥补上述不足，本文结合准循环神经网络（Quasi-Recurrent Neural Network，QRNN）模型对序列数据的处理和预测能力，提出一种基于 QRNN 的网络协议模糊测试用例过滤方法。该方法可以学习到网络协议的结构特征，自动过滤无效测试用例，从而提高网络协议模糊测试的效率。

2 背景知识

2.1 网络协议模糊测试的基本过程

网络协议模糊测试在执行过程中大致要经历 5 个基本阶段[11]，如图 1 所示。

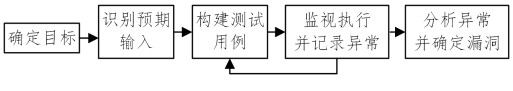

图 1 网络协议模糊测试的基本过程

Fig. 1 Basic process of network protocol fuzzing test

（1）确定目标。确定目标是网络协议模糊测试过程的第一步，只有充分了解协议实体，才能选择合适的网络协议模糊测试工具和方法。

（2）识别预期输入。通常，协议实体在接收外部输入时未能正确处理一些非法数据，导致大多数协议漏洞被利用。如果不能充分分析协议实体所接受的数据规约，就无法进行有针对性的网络协议模糊测试。因此，考虑协议实体所有可能的输入情况对网络协议模糊测试的成功具有重要指导意义。

（3）构建测试用例。构建测试用例是网络协议模糊测试的核心和关键步骤，决定着模糊测试效率。目前测试用例生成策略可以分为两大类：基于变异的策略和基于生成的策略。测试用例集的构建可以采用其中的一种策略，也可以同时采用上述两种策略。

（4）监视执行并记录异常。根据测试用例在协议实体中的执行情况，观察协议实体是否出现异常情况，并记录异常信息。常用的监视记录技术依据原理可以分为两种：基于调试的方法和基于插桩的方法。

（5）分析异常并确定漏洞。确定可利用性是网络协议模糊测试过程中的最后一步，旨在分析协议实体出现异常的位置与原因，并确定漏洞是否被利用。

2.2 基于变异的网络协议模糊测试

基于变异的网络协议模糊测试首先捕获网络中正常通信的协议报文，然后根据变异策略将协议报文中的某些字段更改为非法字段，生成畸形的测试用例，对协议实体进行模糊测试[12]。该方法不要求提前对被测的网络协议进行学习研究，只需针对特定的协议实体，截取通信过程中的网络流量，以比特或字节为单位，根据变异策略修改字段值。这种方法节省了大量的人力，对测试人员的网络协议专业知识要求不高。测试用例经过变异后包含了一些非法值，因此该方法更容易触发协议实体中的不安全函数，但是协议实体可能会拒绝接收含有非法值的测试用例，最终导致生成的测试用例接收率较低，从而影响基于变异的网络协议模糊测试效率。

3 基于 QRNN 的网络协议模糊测试用例过滤方法

3.1 框架设计

基于 QRNN 的网络协议模糊测试用例过滤方法的框架如图 2 所示。

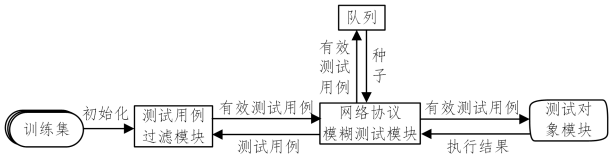

图 2 基于 QRNN 的网络协议模糊测试用例过滤方法的框架图

Fig. 2 Framework of testcase filtering method based on QRNN for network protocol fuzzing test

（1）测试用例过滤模块。该模块基于 QRNN 模型实现，经过初始化后具备预测和过滤的功能。该模块接收从网络协议模糊测试模块中输出的测试用例，并对该测试用例的有效性进行预测，从而过滤无效测试用例，仅将有效测试用例返回给网络协议模糊测试模块。

(2)网络协议模糊测试模块。该模块为 AFL_{NET}，其原型为 AFL[13]，是第一个针对协议实现的灰盒模糊测试工具，主要负责：1)变异协议报文，该模块采用的变异策略包括比特反转、算术加减、特殊内容替换和字典替换等[14]；2)向测试对象模块输入有效测试用例；3)记录测试用例在协议实体中的执行结果。协议实体执行测试用例后，根据执行结果，将覆盖协议实体中新的代码路径、执行后造成协议实体发生新的状态转换、超时或崩溃的测试用例保存进队列。

(3)测试对象模块。该模块为协议实体，主要负责执行预测结果为有效的测试用例，并将执行结果返回给网络协议模糊测试模块。

3.2 QRNN 模型

序列数据的一大特点是前后数据之间具有关联性，而协议报文就是一种序列数据。除此之外，协议报文中的前后数据还具有长程依赖关系[15]，即在一个很长的序列里，开头数据可以对结尾数据产生影响。

循环神经网络(Recurrent Neural Network，RNN)是深度学习中的一种模型，相比其他神经网络，它更擅长处理和预测序列数据。但简单的 RNN 很难建立长时间间隔的状态之间的依赖关系，导致前面时刻的输入对后面时刻的输出影响越来越小。而作为 RNN 的衍生结构，长短期记忆(Long Short-Term Memory，LSTM)[16]网络可以缓解长程依赖问题，符合协议报文的实际特性，但需要引入额外的状态，实现复杂。因此，综合考虑长程依赖问题和训练效率，本文选择 QRNN 作为测试用例过滤模块的实现模型。相比 LSTM 网络，QRNN 训练参数少，结构简单，并且具有良好的并行计算能力，更适用于处理长序列数据。在序列长度一致的条件下，QRNN 可以达到与 LSTM 相近的准确度，但速度是 LSTM 的 2～17 倍，并且序列越长，速度提升越快。

QRNN 由卷积层和池化层两部分组成，其结构如图 3 所示[17]。卷积层用于提取输入特征以及对门函数进行卷积处理。对于一个含有噪音的输入序列 $\boldsymbol{X}=[x_1,x_2,\cdots,x_T]$，经过滤波器个数为 m 的卷积层运算后，得到长度仍为 T 的含噪音序列 $\boldsymbol{Z}=[z_1,z_2,\cdots,z_T]$。其中，序列 $\boldsymbol{X}$ 中每个向量的维度为 n，序列 $\boldsymbol{Z}$ 中每个向量的维度为 m。如果滤波器的宽度设置为 k，则 t 时刻 z_t 的范围为 $x_{t-k+1}\sim x_t$。卷积层的表达式如下所示：

$$\boldsymbol{Z}=\tanh(\boldsymbol{W}_z*\boldsymbol{X}) \tag{1}$$

$$\boldsymbol{F}=\sigma(\boldsymbol{W}_f*\boldsymbol{X}) \tag{2}$$

$$\boldsymbol{O}=\sigma(\boldsymbol{W}_o*\boldsymbol{X}) \tag{3}$$

其中，$\boldsymbol{W}_z,\boldsymbol{W}_f,\boldsymbol{W}_o$ 为卷积滤波器，σ 为激活函数，$*$ 表示滤波器在序列维度上的卷积运算。本文实验设置滤波器的宽度 $k=2$，此时，QRNN 的卷积层表达式如下所示：

$$\boldsymbol{z}_t=\tanh(\boldsymbol{W}_z^1\boldsymbol{x}_{t-1}+\boldsymbol{W}_z^2\boldsymbol{x}_t) \tag{4}$$

$$\boldsymbol{f}_t=\sigma(\boldsymbol{W}_f^1\boldsymbol{x}_{t-1}+\boldsymbol{W}_f^2\boldsymbol{x}_t) \tag{5}$$

$$\boldsymbol{o}_t=\sigma(\boldsymbol{W}_o^1\boldsymbol{x}_{t-1}+\boldsymbol{W}_o^2\boldsymbol{x}_t) \tag{6}$$

池化层用于提取卷积层输出的特征信息，减少特征数目。池化层的计算是在动态平均池化的基础上增加了输出门和遗忘门。这样的结构类似于 LSTM 的门结构，池化层的表达式如下所示：

$$\boldsymbol{c}_t=\boldsymbol{f}_t\odot\boldsymbol{c}_{t-1}+\boldsymbol{i}_t\odot\boldsymbol{z}_t \tag{7}$$

$$\boldsymbol{h}_t=\boldsymbol{o}_t\odot\boldsymbol{c}_t \tag{8}$$

其中，$\boldsymbol{f},\boldsymbol{i},\boldsymbol{o}$ 分别表示遗忘门、输入门和输出门，$\boldsymbol{c}_{t-1}$ 表示 t 时刻的记忆单元状态，$\boldsymbol{h}$ 和 $\boldsymbol{c}$ 的初始状态为 0，$\odot$ 表示向量之间的点乘运算。QRNN 可以实现数据的并行计算，并且在输出时有效地利用了输入序列的顺序信息[18]。

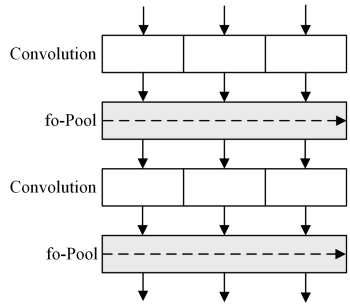

图 3 QRNN 结构图

Fig. 3 QRNN structure diagram

3.3 测试用例过滤模型初始化

测试用例过滤的目的是从大量随机变异出的测试用例中过滤出有效性较低的测试用例[19]，因此，首先需要确定测试用例有效性的标准。本文将以下 4 种类型的测试用例定义为有效测试用例：

(1)在协议实体中执行后覆盖新的代码路径的测试用例；

(2)在协议实体中执行后造成协议实体发生新的状态转换的测试用例；

(3)在协议实体中执行后造成协议实体超时，即长时间无响应的测试用例；

(4)在协议实体中执行后造成协议实体崩溃，即有可能触发漏洞的测试用例。

其他测试用例均视作无效测试用例。

由于网络协议种类繁多，不同协议的报文也千差万别，本文旨在设计一种通用的测试用例过滤方法，以适用于各种不同的网络协议。因此，本文按照字节对测试用例进行划分，形成字节流向量 $\boldsymbol{X}=\langle\boldsymbol{x}_1',\boldsymbol{x}_2',\cdots,\boldsymbol{x}_n'\rangle$，其中，$\boldsymbol{x}_i'=\boldsymbol{x}_i+1$，$\boldsymbol{x}_i$ 是测试用例的第 i 个字节且 $\boldsymbol{x}_i\in\{0,1,\cdots,255\}$，$n$ 是测试用例的总字节数。输出为一个类别 $y\in\{0,1\}$，其中 $y=0$ 表示无效测试用例，$y=1$ 表示有效测试用例。

由于 QRNN 模型的输入特性，训练集中的测试用例需要进行归一化处理，即统一所有测试用例的长度。设训练集 $\boldsymbol{S}=\langle\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n\rangle$，其中测试用例 $\boldsymbol{X}_m$ 具有最大长度 s，则对除 $\boldsymbol{X}_m$ 之外的所有测试用例均要进行尾部补 0 操作，使其长度达到 s，以满足 QRNN 模型的输入要求。

将测试用例划分形成的字节流向量 $\boldsymbol{X}=\langle\boldsymbol{x}_1',\boldsymbol{x}_2',\cdots,\boldsymbol{x}_T'\rangle$ 中的每一个元素都看作是 QRNN 模型中一个时刻的输入，并得到不同时刻的隐藏状态 $\boldsymbol{h}_1,\boldsymbol{h}_2,\cdots,\boldsymbol{h}_T$。本文将最后时刻的隐藏状态 $\boldsymbol{h}_T$ 看作整个测试用例的表示，并输入给分类器 $g(\cdot)$ 进行有效性预测[20]。基于上述的输入和输出，本文设计的测试用例过滤模型如图 4 所示。

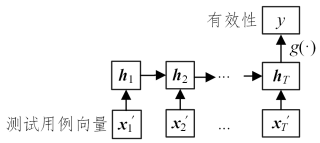

图 4 测试用例过滤模型

Fig. 4 Testcase filtering model

3.4 训练集获取

在生成模型之前，需要先进行训练集的获取。由于测试用例被分为有效和无效两类，QRNN 模型生成问题可以转化为一个二分类问题。二分类问题一般通过训练正负样本来生成模型。因此，本文将有效测试用例视作正样本，将无效测试用例视作负样本。

AFL_{NET} 在实际工作中只会保存有效测试用例，而模型的生成除了正样本外，还需要负样本。因此，为了获取训练集，需要修改 AFL_{NET} 的源码。AFL_{NET} 将测试用例送入协议实体中执行后，根据执行结果，将覆盖新的代码路径、产生新的状态转换、超时或崩溃的测试用例保存进有效测试用例目录，将其他的测试用例保存进无效测试用例目录。针对无状态网络协议，获取训练集的基本流程如图 5 所示。

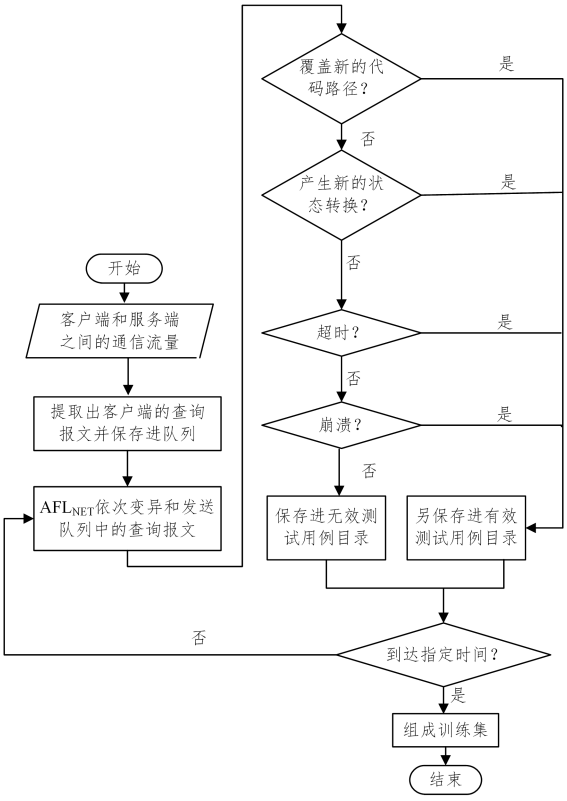

图 5 针对无状态网络协议的训练集获取流程

Fig. 5 Training set acquisition process for stateless network protocol

(1)嗅探客户端和服务端之间的通信流量并从中提取出客户端的查询报文，保存进队列；

(2)AFL_{NET} 依次变异和发送队列中的查询报文；

(3)若变异产生的测试用例在协议实体中执行后覆盖新的代码路径，造成协议实体产生新的状态转换、超时或崩溃，则另保存进有效测试用例目录，否则只保存进无效测试用例目录；

(4)重复步骤 2 和步骤 3，直到指定时间；

(5)将有效测试用例和无效测试用例作为有标签训练样本组成训练集。

3.5 训练集预处理

分类学习问题都有一个共同的基本假设，即不同类别的训练样本数量相当。如果不同类别的训练样本数量相差较大，属于极端不平衡，则会严重影响分类学习的效果。假设在二分类学习中，负样本有 998 个，正样本只有 2 个，那么只需返回一个永远将新样本预测为负样本的分类器，就能达到 99.8%的精度。然而，这样的分类器往往没有价值，因为它不能预测出任何正样本。

一般情况下，绝大多数变异产生的测试用例在测试对象中的执行结果都是无效的，导致训练集中绝大多数都是负样本，而正样本只占少数[21]。因此，可以认为网络协议模糊测试用例过滤是一个类不平衡问题，而欠采样对于类不平衡问题是一个较好的解决方法。

欠采样方法通过减少多数类样本的数量，调整训练集中不同类样本的分布情况，降低类间的不平衡比例，以适合分类器进行学习[22]。而随机欠采样是一种最简单的欠采样方法，但可能会导致多数类样本中重要信息的丢失。因此，本文采用一种自定义的半随机欠采样方法，通过事先设置多数类样本和少数类样本的数量比例，以及考虑到样本字节长度对模型生成速度的负面影响和少数类样本的最大字节长度，在保留少数类样本不变的情况下，根据多数类样本中不同字节长度样本的分布比例，随机选择多数类样本，从而达到平衡的效果。

3.6 测试用例过滤模型搭建

测试用例过滤模型搭建主要包括数据初始化、前向计算和结果输出。具体算法如算法 1 所示。

算法 1 测试用例过滤模型搭建算法 Build_Model

输入：输入大小 input_size、隐藏层单元大小 hidden_size、输出大小 output_size、隐藏层层数 layer_num 和双向标志 bidirectional

输出：测试用例过滤模型

```
1. class Classifier(Module):
2.     def init(input_size,hidden_size,output_size,layer_num=1,bidirectional=False):
3.         embedding=Embedding(input_size,hidden_size)//初始化嵌入层
4.         qrnn=QRNN(hidden_size,hidden_size,layer_num,dropout=0.4)//初始化 QRNN
5.         linear=Linear(hidden_size * bidirectional,output_size)//初始化线性层
6.         def init_hidden(batch_size):
7.         hidden=zeros(layer_num * bidirectional,batch_size,hidden_size)//初始化隐藏层
8.     def forward(input):
9.         input=input.t()//转置输入
10.        batch_size=input.size(1)//获取 batch_size
11.        hidden=init_hidden(batch_size)
12.        embedding=embedding(input)
```

```
output,hidden=qrnn(embedding,hidden)//搭建 QRNN 模型
if bidirectional://双向网络
    hidden=cat([hidden[-1],hidden[-2]],dim=1)
else://单向网络
    hidden=hidden[-1]
output=linear(hidden)
return output//返回预测结果
```

3.7 测试用例预测及过滤

传统的网络协议模糊测试用例过滤方法中，AFL_{NET}需要通过执行测试用例来预测其有效性并进一步过滤无效测试用例。而本文方法通过搭建测试用例过滤模型，接收从 AFL_{NET}中输出的测试用例，并对其有效性进行预测，从而过滤无效测试用例，仅将有效测试用例返回给 AFL_{NET}。具体算法如算法 2 所示。

算法 2 测试用例预测及过滤算法 Predict_Filter

```
输入:测试用例过滤模型
输出:有效测试用例集合
classify()//预测测试用例有效性
    pModule=PyImport_ImportModule("predict")//导入 predict 文件
    pFunc = PyObject _ GetAttrString ( pModule," predict _ testcase")//搜索 predict_testcase 函数
    PyObject _ CallFunction ( pFunc," O", pRet )//调 用 predict _ testcase 函数
    Py_DECREF(pFunc)//释放 pFunc
    Py_DECREF(pModule)//释放 pModule
if(label)//若预测结果为有效
    valid_testcase++//计数有效测试用例
    fault=run_target()//AFLNET执行有效测试用例
    queued_discovered+=save_if_interesting(fault)//判断预测结果的准确性并保存有效测试用例
    return 0//继续变异
else//若预测结果为无效
    return 0//过滤无效测试用例并继续变异
```

4 实验验证

4.1 实验设置

AFL_{NET}是第一个针对协议实现的灰盒模糊测试工具，而本文提出的基于 QRNN 的网络协议模糊测试用例过滤方法将与 AFL_{NET}进行对比，两种方法采用的测试用例均由相同的初始有效测试用例以相同的变异策略生成。作为一种代表性的无状态协议，几乎每个互联网连接都是始于 DNS 查询[23]，因此，本文实验选取 DNS 协议实体 BIND 9 作为测试对象。BIND 9 是目前应用广泛、功能全面的 DNS 系统之一，其基本信息如表 1 所列。

表 1 测试对象的基本信息

Table 1 Basic information of test object

协议类型	测试对象	版本
DNS	BIND 9	v9.9.6

硬件配置为 Intel(R) Xeon(R) Gold 6139 CPU @ 2.30 GHz，376 GB 的 Ubuntu 18.04 服务器。

4.2 评价标准

网络协议模糊测试效率指相同时间内覆盖协议实体的总路径数或触发协议实体崩溃的测试用例数，数量越多则效率越高。本文实验选择网络协议模糊测试效率作为评价基于 QRNN 的网络协议模糊测试用例过滤方法的标准，效率越高，证明方法越有效。

需要说明的是，在模糊测试之前，初始队列中通常需要存储一定数量的有效测试用例，以尽可能完全覆盖测试对象的代码分支，从而挖掘出足够数目的漏洞。而在网络协议模糊测试中，由于无法针对协议实体准备足够数量的有效测试用例，通常难以在一定时间内多次触发协议实体崩溃，因此，为了更直观地评价所提方法的有效性，实验只比较网络协议模糊测试效率中相同时间内覆盖协议实体的总路径数这一项。

4.3 训练集获取和预处理

针对 BIND 9，AFL_{NET}运行半小时以获取训练集。其中，有效测试用例数为 599，最大字节长度为 17 394；无效测试用例数为 4349，最大字节长度为 506 274。测试用例中不同字节长度样本的分布比例如图 6 所示。由此可见，绝大多数无效测试用例的字节长度都小于 50 000。

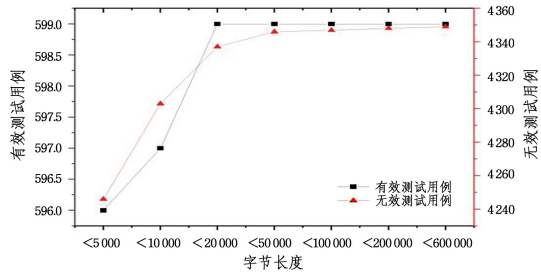

图 6 测试用例中不同字节长度样本的分布比例

Fig. 6 Distribution proportion of samples with different byte lengths in testcases

训练集中的有效测试用例和无效测试用例均为 DNS 查询报文，其格式如图 7 所示。1)标识占两个字节，同一个问题的查询和响应标识必须相同；2)标志占两个字节，包含 QR，opcode，AA，TC，RD，RA，(zero)和 rcode 8 个标志位；3)问题数、资源记录数、授权资源记录数和附加资源记录数均占两个字节，对于查询报文，通常问题数为 1，其他 3 项为 0；4)查询问题由查询名、查询类型和查询类 3 部分组成。查询名长度不定，一般为待查询的域名(反向查询时为 IP 地址)；查询类型占两个字节，常用的有 A，NS 和 PTR；查询类占两个字节，通常为 1，指互联网地址。

图 7 DNS 查询报文格式

Fig. 7 DNS query message format

考虑到训练集中有效测试用例的类型有限，因此，特将 BIND 9.9.6 部分漏洞的公开 PoC 按照 DNS 查询报文的格式处理后作为有效测试用例添加进训练集中，其字节长度均

小于 50000。漏洞的基本信息如表 2 所列。

表 2 漏洞的基本信息

Table 2 Basic information of vulnerabilities

标识	等级
CVE-2015-5477	高危
CVE-2015-5722	高危
CVE-2015-8000	低危
CVE-2016-1285	低危
CVE-2016-2775	低危
CVE-2016-2776	高危
CVE-2016-8864	低危
CVE-2020-8617	低危

设有效测试用例和无效测试用例的数量比例约为 1∶1，同时考虑到样本字节长度对模型搭建速度的负面影响和有效测试用例的最大字节长度，在保留有效测试用例不变的情况下，根据无效测试用例中不同字节长度样本的分布比例，随机选择无效测试用例数，组成如图 8 所示的类平衡训练集。其中，有效测试用例数为 607，最大字节长度为 17394；无效测试用例数为 598，最大字节长度为 37571。

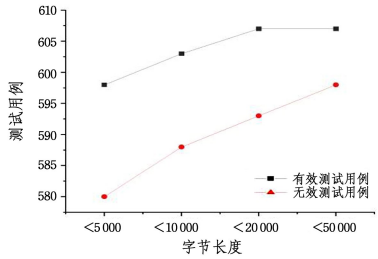

图 8 类平衡训练集的组成

Fig. 8 Composition of class-balanced training set

4.4 参数选取

参数选取不仅决定着整个模型训练过程中的运行速度，而且还会影响模型的分类性能。实验发现，hidden_size 和 dropout 这两种参数的选取对模型的运行速度和分类性能具有重要意义。其中，较大的 hidden_size 会增加模型的复杂程度，降低模型的泛化能力；较小的 hidden_size 会导致特征提取效果不理想。选取合适的 dropout 可以有效地防止模型过拟合，解决特征提取不全面的问题。

为了获得具有最优分类性能的模型，实验分析了在选取不同 dropout 的情况下，hidden_size 对模型分类性能的影响，实验结果如图 9 所示。其中 hidden_size 分别选取 2，4，6 和 8，而 dropout 则分别选取 0.2，0.3，0.4 和 0.5。

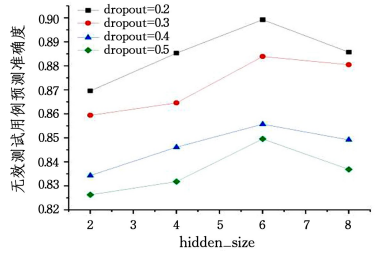

图 9 hidden_size 和 dropout 对模型分类性能的影响

Fig. 9 Influence of hidden_size and dropout on model classification performance

由图 9 所示的实验结果可知，选取不同的 hidden_size 和 dropout，模型的分类性能有一定程度的变化。当选取 hidden_size 为 6、dropout 为 0.2 时，模型的分类性能达到最优，此时，无效测试用例预测准确度为 89.92%。

测试用例形成字节流向量后，各分量的取值范围为[0，256]，因此，设置输入的大小为 257；测试用例的预测结果只分为有效和无效两类，因此，设置输出的大小为 2；隐藏层层数设置为 2，最大迭代次数设置为 50，双向标志设置为 True，其余参数采用默认设置。此外，采用 Adam 优化算法和 CrossEntropyLoss 函数，以加快模型的收敛速度，进一步优化模型的分类性能。具体的参数设置如表 3 所列。

表 3 模型参数设置

Table 3 Model parameters setting

参数	数值
input_size	257
hidden_size	6
layer_num	2
bidirectional	True
dropout	0.2
output_size	2
epoch_num	50
use_gpu	False
learning_rate	0.001

4.5 实验结果与分析

以 BIND 9 作为测试对象，测试时间设置为 3h，初始队列中的有效测试用例数设置为 1，AFL_{NET} 与本文方法的测试结果如表 4 所列。

表 4 测试结果

Table 4 Test results

测试方法	AFL_{NET}	QRNN 模型
测试时间/h	3	3
总路径数	1275	2360
总测试用例数	128000	146000
无效测试用例执行数	126726	55830
单个测试用例执行\预测时间/s	0.0312	0.0026

由表 4 可知，BIND 9 执行单个测试用例所耗费的时间大约是 0.0312 s，而 QRNN 模型预测单个测试用例所耗费的时间大约是 0.0026 s。对于 AFL_{NET}，BIND 9 执行的无效测试用例数高达约 99%，浪费了约 36.61%的测试时间；而对于本文方法，BIND 9 执行的无效测试用例数大约仅占 38.24%，避免了一半以上的无效执行。

由表 4 可知，AFL_{NET} 触发在 3h 的测试时间内覆盖 BIND 9 的总路径数为 1275，而本文方法在 3 h 的测试时间内覆盖 BIND 9 的总路径数为 2360。在测试时间相同的条件下，相比 AFL_{NET}，本文方法覆盖协议实体的总路径数增加约 85.1%，即在网络协议模糊测试效率方面，本文方法远高于 AFL_{NET}，具有明显的优势。由实验可知，本文提出的基于 QRNN 的网络协议模糊测试用例过滤方法达到了预期的测试效果，可自动过滤无效测试用例以减少无效执行时间，有利于 AFL_{NET} 专注有效测试用例的生成，提升有效测试用例生成效率，从而进一步增加单位时间内覆盖 BIND 9 的总路径数，网络协议模糊测试效率显著提高，验证了本文本文方法的有效性。

结束语 本文研究网络协议模糊测试用例过滤问题，提出了一种基于神经网络的网络协议模糊测试用例过滤方法。通过 QRNN 模型学习网络协议样本得到协议的结构特征，预测过滤无效测试用例。实验结果表明，模型预测无效测试用例的准确度可达到约 90%，网络协议模糊测试效率也显著提高，从而验证了本文方法的有效性。本文的不足之处在于被测网络协议类型单一和有效测试用例的误报率高，下一步将研究有状态网络协议以及如何提高神经网络模型预测有效测试用例的准确度，以进一步提升网络协议模糊测试效率。

参 考 文 献

[1] LI J, ZHAO B, ZHANG C. Fuzzing: asurvey[J]. Cybersecurity, 2018, 1(1): 1-13.

[2] COHEN M B, SNYDER J, ROTHERMEL G. Testing across configurations: implications for combinatorial testing[J]. ACM SIGSOFT Software Engineering Notes, 2006, 31(6): 1-9.

[3] LIANG H, PEI X, JIA X, et al. Fuzzing: State ofthe art[J]. IEEE Transactions on Reliability, 2018, 67(3): 1199-1218.

[4] PEACHTEC. Peach[EB/OL]. (2017-10-06) [2021-04-17]. http://www.peachfuzzer.com/products/peach-platform.

[5] Beyond Security. beSTORM[EB/OL]. (2021-04-17) [2021-04-17]. https://beyondsecurity.com/solutions/bestorm.html.

[6] PHAM V T, BÖHME M, ROYCHOUDHURY A. AFL$_{Net}$: a greybox fuzzer for network protocols[C]// 2020 IEEE 13th International Conference onSoftware Testing, Validation and Verification(ICST). IEEE, 2020: 460-465.

[7] LI M L, HUANG H, LU Y L. Test Case Generation Technology Based on Symbol Divideand Conquer Area for Vulnerability Mining[J]. Netinfo Security, 2020, 20(5): 39-46.

[8] GONG W, ZHANG G, ZHOU X. Learn to Accelerate Identifying New Test Cases in Fuzzing[C]// International Conference on Security, Pri-vacy and Anonymity in Computation, Communication and Storage. Cham: Springer, 2017: 298-307.

[9] KARAMCHETI S, MANN G, ROSENBERG D. Improving Grey-Box Fuzzing by Modeling Program Behavior[J]. arXiv: 1811.08973, 2018.

[10] ZONG P, LV T, WANG D, et al. Fuzzguard: Filteringout unreachable inputs in directed grey-box fuzzing through deep learning[C]// 29th Security Symposium (USENIX). 2020: 2255-2269.

[11] ZHANG X, LI Z J. Surveyof Fuzz TestingTechnology[J]. Computer Science, 2016, 43(5): 1-8, 26.

[12] JIANG Y G, CHEN X, LI J B, et al. A FuzzyTest Case Generation Method based on LSTM for S7 Protocol[J]. Computer Engineering, 2021, 47(7): 183-188.

[13] ZALEWSKI M. American fuzzy lop [EB/OL]. (2017-11-05) [2021-04-17]. https://github.com/mirrorer/afl.

[14] LCAMTUF. AFL fuzzing strategies [EB/OL]. (2014-08-08) [2021-04-17]. https://lcamtuf.blogspot.jp/2014/08/binary-fuzzing-strategies-what-works.html.

[15] SCHMIDHUBER J. Gradient Flow in RecurrentNets: the Difficulty of Learning Long-Term Dependencies[M]// Wiley-IEEE Press, 2001.

[16] HOCHREITER S, SCHMIDHUBER J. Long Short-Term Memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[17] BRADBURY J, MERITY S, XIONG C, et al. Quasi-recu-rrent neural networks[J]. arXiv: 1611.01576, 2016.

[18] LOU Y X, YUAN W H, PENG R Q. Speech Enhancement Method Based on Quasi Recurrent Neural Network[J]. Computer Engineering, 2020, 46(4): 316-320.

[19] WANG Y, JIA P, LIU L, et al. A systematic reviewof fuzzing based on machine learning techniques[J]. PLoS ONE, 2020, 15(8): e0237749.

[20] QIU X P. Neural Networks and Deep Learning[M]. Beijing: China Machine Press, 2020.

[21] ZHOU Y H. Research on Network Protocol Vulnerability Mining Method Based on Deep Learning[D]. Chengdu: University of Electronic Science and Technology of China, 2020.

[22] XU L L, CHI D X. Machine learning classification strategy for imbalanced data sets[J]. Computer Engineeringand Applications, 2020, 56(24): 12-27.

[23] BIND 9[EB/OL]. (2004-01-28) [2021-04-17]. https://www.isc.org/bind/.

HU Zhi-hao, born in 1997, postgraduate. His main research interests include network security and fuzzing test.

PAN Zu-lie, born in 1976, Ph.D, professor. His main research interests include network security, vulnerability discovery and computer science.

（责任编辑：柯颖）