doi :10.3969/j.issn.1671-1122.2019.03.004

基于深度学习的浏览器 Fuzz 样本生成技术研究

方勇[1]，朱光夏天[2]，刘露平[2]，贾鹏[2]

（1. 四川大学网络空间安全学院，四川成都 610207；2. 四川大学电子信息学院，四川成都 610065）

摘　要：在众多软件漏洞挖掘的方法中，Fuzz 测试是最为成熟有效的一种。而传统的 Fuzz 测试普遍存在挖掘深度不足、样本没有指向性等问题。针对该问题，文章提出一种使用长短期记忆网络（Long Short Term Memory, LSTM）引导生成浏览器 Fuzz 所需的样本集的框架。该框架包含样本生成和模糊测试两个部分。首先，对样本进行预处理，将样本解析为向量送入神经网络中学习。其次，待神经网络学习完成后，利用学习完成的网络生成样本，并利用传统变异策略将生成的样本进行变异，构成测试集。最后，使用测试集作为输入进行浏览器 Fuzz 测试。为验证该框架的有效性，对 LSTM 网络的学习结果、生成样本结果和 Fuzz 结果进行了统计与分析。实验证明，该框架能满足浏览器 Fuzz 生成的需求，并克服了传统浏览器 Fuzz 中样本挖掘深度不足、指向性弱的问题，适合针对某一类或某几类浏览器漏洞的挖掘。

关键词：浏览器 Fuzz；深度学习；样本生成；LSTM 神经网络；文件向量化

中图分类号：TP309　**文献标识码：**A　**文章编号：**1671-1122（2019）03-0026-08

中文引用格式：方勇，朱光夏天，刘露平，等 · 基于深度学习的浏览器 Fuzz 样本生成技术研究 [J]· 信息网络安全，2019，19（3）：26–33·

英文引用格式：FANG Yong, ZHU Guangxiatian, LIU Luping, et al· Research on Browser Fuzz Sample Generation Technology Based on Deep Learning[J]· Netinfo Security, 2019, 19(3): 26–33·

http://www.ixueshu.com

Research on Browser Fuzz Sample Generation Technology Based on Deep Learning

FANG Yong[1], ZHU Guangxiatian[2], LIU Luping[2], JIA Peng[2]

(1. *College of Cybersecurity*, *Sichuan University*, *Chengdu Sichuan* 610207, *China*; 2. *College of Electronics and Information*, *Sichuan University*, *Chengdu Sichuan* 610065, *China*)

Abstract: Fuzz testing is one of the most mature and effective methods among the approaches used to mine vulnerabilities for modern software. However, traditional Fuzz testing generally have some problems, such as limited depth of exploring code space or lacking of directivity in generating samples. To alleviate these issues, a kind of framework

收稿日期：2019-1-10

基金项目：国家重点研发计划 [2017YFB0802900]

作者简介：方勇（1966—），男，四川，教授，博士，主要研究方向为信息安全理论与应用、网络攻防及网络行为监管技术；朱光夏天（1993—），男，湖北，硕士研究生，主要研究方向为 Windows 安全、漏洞挖掘与利用；刘露平（1988—），男，四川，博士研究生，主要研究方向为二进制安全、漏洞挖掘；贾鹏（1988—），男，河南，博士研究生，主要研究方向为病毒传播动力学、二进制安全、恶意代码分析。

通信作者：朱光夏天 756662392@qq.com

was proposed to generate samples of browsers by making use of long short term memory (LSTM) network. The framework consists two components: sample generating and Fuzz testing. Firstly, the sample are encoded into vectors which are much easier to implement in LSTM network. This process is called file preprocessing. After finishing the learning period, the network will generate a mound of samples as test set. Then test set will be generated by mutating samples based on traditional mutation strategies. Finally, the test set will be feed into the browser for Fuzz testing. In order to verify the effectiveness of the framework, the learning results, generating sample results and Fuzz results of LSTM network have been analyzed statistically. It is proofed that the proposed framework could satisfy the needs of browser Fuzz generation and overcome the difficulties of insufficient mining depth and lack of directivity in generating samples in traditional browser Fuzz, which was suitable for mining one or several browser vulnerabilities.

Key words: browser Fuzz; deep learning; sample generation; LSTM neural network; file vectorization

0 引言

随着信息社会的不断发展，软件安全已经成为信息安全的重要基础保障。软件安全漏洞是对信息安全的重要威胁，对软件进行安全测试和漏洞挖掘是提高软件安全性的重要措施。Fuzz 技术作为一种有效的程序测试技术被广泛应用于软件安全测试和漏洞挖掘 [1]。

Fuzz 技术按样本集获取方式划分为两类：基于生成的 Fuzz 和基于变异的 Fuzz。基于生成的 Fuzz 技术指通过生成获得大量的符合程序输入格式的样本，基于变异的 Fuzz 技术指通过变异已有的样本得到大量新的输入样本。

传统的基于生成的 Fuzz 测试中通常采用随机策略生成样本集。在已有的数据集中随机选择结构完整的代码语句插入待输出的文件，并按照一定概率对语句进行置换、删除、插入等操作。这种大量的随机操作使得生成的样本能快速发现软件中的浅层漏洞。但在挖掘深层次漏洞时效率较低，其主要原因有：

1）Fuzz 样本集检测广度有余而深度不足。传统 Fuzz 策略生成的样本大多具有较高的代码重复率，对出现概率较高的函数、输入点进行大量且重复的试探。

2）生成样本集时具有一定的盲目性，仅依靠已知数据集内的随机选择和有限的变异来生成特殊样本，没有足够的指向性，对脆弱点的挖掘力度不够。

3）样本载入被测试软件后，只能反馈成功或者失败的结果，而对失败原因没有任何反馈，造成大量类似样本被清除后又被重新生成，无法有效利用测试结果。

近年来漏洞挖掘与深度学习技术结合产生了许多成果。在程序分析和漏洞挖掘方面，PANG[2] 等人先后提出了基于支持向量机（Support Vector Machine，SVM）集成学习的软件组件早期漏洞的识别方法和利用 *N*-Gram 模型和统计特征选择来预测漏洞的方法 [3]，将深度学习应用于二进制漏洞预测与恶意代码识别。在国内，文伟平 [4] 等人也取得了同一领域的成果。GODEFROID[5] 等人首次将深度学习应用于漏洞挖掘，他们使用 seq2seq 架构来生成 PDF 样本。WANG[6] 等人则利用深度学习网络解析文件结构，引导生成用于 Fuzz 的种子文件。

为解决传统的基于生成的 Fuzz 中出现的样本深度不足、指向性不够等问题，本文提出一种使用神经网络指导 Fuzz 样本集的生成方向的框架，期望生成更有方向性与针对性的 Fuzz 样本，使漏洞挖掘效率更高。

1 基于深度学习的 Fuzz 样本生成框架

基于深度学习的 Fuzz 样本生成框架的结构如图 1 所示，分为样本预处理、神经网络编码、神经网络解码、样本生成 4 个步骤。样本预处理操作包括元素解析与

http://www.ixueshu.com

样本向量化。神经网络编码对输入向量的解析、学习，最后通过全连接层耦合输出，获得输出向量。同时为了避免神经网络过拟合，对网络进行 Dropout 操作。神经网络解码使用 softmax 解码输出的向量、拆分提取输出元素代表的向量，得到的向量与预处理中得到的字典比对，选取最大似然向量对应的文本或标签，与拆分出的位置和频率信息组合作为输出。样本生成使用解码后的输出对空文件进行填充，生成测试样本。

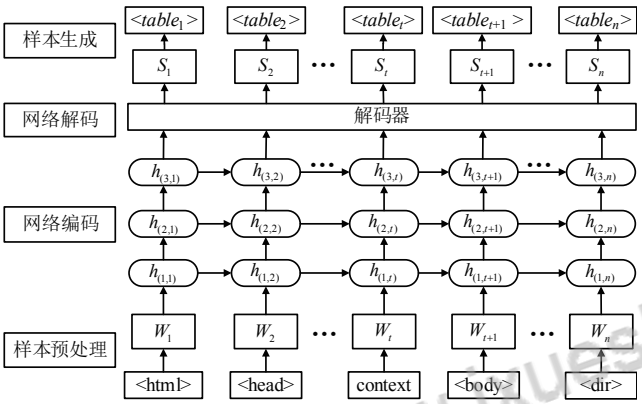


图 1 浏览器 Fuzz 测试样本集生成框架

1.1 样本预处理

与普通的文本语句不同，html 文件是具有结构信息的文件，若不对其进行处理而直接输入神经网络，则神经网络无法识别其结构信息，对生成满足 html 格式的样本不利。故需要对样本进行预处理，将其解析为带有位置与属性概率信息的向量，以便神经网络处理与学习[7]。样本预处理操作分为元素解析与数据向量化。元素解析的目的为分离样本中的标签与文本，并对标签与文本的出现位置与出现频率分别进行统计。然后将统计结果存入字典中，在解析新样本时同步更新字典中标签与文本的属性和位置等信息。向量化的目的是将各个标签与文本以词为单位转化为向量，与元素解析中的统计结果一起组成神经网络的输入向量。

1）元素解析

样本中的标签与文本都是组成样本的元素，元素解析以统计样本中元素信息为目的，具体过程如图 2 所示。

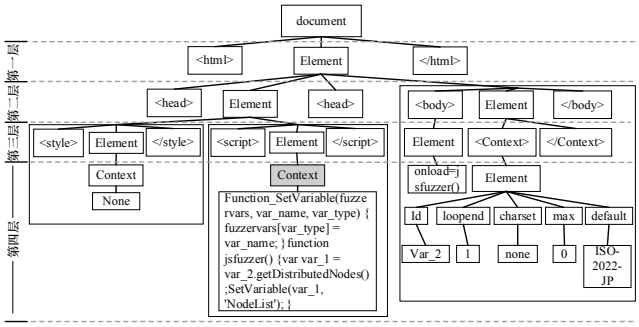


图 2 新输入样本解析样例

新输入的样本被解析成标签和文本的步骤如下：

（1）确定并记录第一层标签 html 及标签属性、文本信息。

（2）确定并记录第二层标签 head、body 及标签属性和文本信息。

（3）确定并记录第三层标签 style、script、context 及第二层标签的标签属性。

（4）确定并记录第四层标签及第三层标签属性，同时确定并记录文本，第四层标签有可能为空。

样本解析结束后，整合该样本标签与文本的相关数据，更新标签与文本的类型、出现位置、频率等数据。

2）向量化

在统计完某批次输入样本中的所有元素后，将元素信息使用 one-hot 编码器编码，并将编码后的向量与元素一一对应存入字典，在解码时使用输出的向量与字典中的向量进行最大相似性查找，从而将向量解码为文本。在标签、标签属性、文本中，N 表示自身数据信息，L 表示相对位置信息，P 表示其出现的概率信息。标签、标签属性、文本的值组成的集合分别用 N_t、N_e、N_c 表示，位置信息分别用 L_t、L_e、L_c 表示，概率信息分别用 P_t、P_e、P_c 表示。取值范围分别为：$N_t \in$(html,head,address,body,script…)；$N_e \in$(class,style,title,onload,dir…)；$N_c \in$(function(),try(),var()…)；$L \in [1,2,3\cdots]$；$P \in (0,1)$。

N_t、N_e、N_c 集合均在接收样本的过程中增大并将自身每一个元素的概率记录为 P_t、P_e、P_c，每接收到一个新的标签或者文本便更新其概率。L_t 为标签出现在文件中的相对位置，概率 P 的计算如公式（1）所示。

http://www.kuaishu.com

$$P=\frac{\text{该元素出现在此处的次数}}{\text{此处出现元素的次数}+1} \quad (1)$$

公式（1）中“此处出现元素的次数”将在每次概率迭代之后更新。各向量的定义与表达式如表 1 所示。表示标签 i 所含信息的标签 $\boldsymbol{N}_{tsi}$ 由标签词向量 $\boldsymbol{N}_{twi}$ 与标签属性词向量 $\boldsymbol{N}_{ewi}$ 组成。其中，$\boldsymbol{N}_{ti}$、$\boldsymbol{N}_{ei}$ 均为标签 i 的标签内容和标签属性使用类 one-hot 编码转化得到的向量，该向量与位置信息 L_{ti}、L_{ei} 和概率信息 P_{ti}、P_{ei} 连接，得到能唯一表示该标签内容与标签属性的词向量 $\boldsymbol{N}_{twi}$、$\boldsymbol{N}_{ewi}$。同理，$\boldsymbol{N}_{ci}$ 为文本 i 转化而来的向量，与文本 i 的位置信息 L_{ci} 和概率信息 P_{ci} 连接，得到能唯一描述该文本内容的文本词向量 $\boldsymbol{N}_{cwi}$。将文本词向量与离其最近的标签词向量连接，得到文本向量 $\boldsymbol{N}_{csi}$。在学习过程中，标签向量或文本向量作为神经网络的输入。

表 1 各向量定义与其表达式

向量定义	表达式
标签词向量	$\boldsymbol{N}_{twi}=(\boldsymbol{N}_{ti},L_{ti},P_{ti})$
标签属性词向量	$\boldsymbol{N}_{ewi}=(\boldsymbol{N}_{ei},L_{ei},P_{ei})$
文本词向量	$\boldsymbol{N}_{cwi}=(\boldsymbol{N}_{ci},L_{ci},P_{ci})$
标签向量	$\boldsymbol{N}_{tsi}=(\boldsymbol{N}_{twi},\boldsymbol{N}_{ewi})$
文本向量	$\boldsymbol{N}_{csi}=(\boldsymbol{N}_{twi},\boldsymbol{N}_{wi})$
标签向量集	$\sum\boldsymbol{N}_t=(\boldsymbol{N}_{ts1},\boldsymbol{N}_{ts2},\boldsymbol{N}_{tsk})$
文本向量集	$\sum\boldsymbol{N}_c=(\boldsymbol{N}_{cs1},\boldsymbol{N}_{cs2},\boldsymbol{N}_{csk})$
输入向量集	$\sum\boldsymbol{N}=(N_t,N_c)$

由于使用 one-hot 编码的向量维度一致，故插入位置信息与概率信息后的位置依旧一致。所以不需要对向量额外进行切片与填充操作以保证输入向量长度相等 [8]。

所有的标签向量构成的集合为标签向量集，所有的文本向量构成的集合为文本向量集，所有输入的向量集合 $\sum\boldsymbol{N}$ 由标签向量集和文本向量集组成。在样本处理中，标签向量集、文本向量集、输入向量集均保持更新，但不作为输入。在生成过程中，这 3 个集合中的数据将作为样本生成的参考数据。

1.2 编码层

本文选择 LSTM 作为编码网络。成熟的递归神经网络（Recurrent Neural Network，RNN）已经被广泛用于如文本预测 [9, 10]、图像识别 [11] 等领域。而在处理结构化文档与文本预测方面，LSTM 更有优势。作为 RNN 更高效稳定的变形，LTSM 网络已经能完成自动生成网页 [12, 13]、语义分析与预测 [14] 等工作。同时还避免了神经网络在深度学习中的梯度消失问题和梯度爆炸问题 [14]。

在编码层，向量将输入到神经网络细胞中。LSTM 的关键为细胞状态，输入、输出、遗忘等操作均围绕细胞状态进行。其典型细胞状态更新操作如图 3 所示。新数据传入时细胞状态以当前结构进行一次迭代，而迭代这一功能则由数个“门”结构完成 [8]。

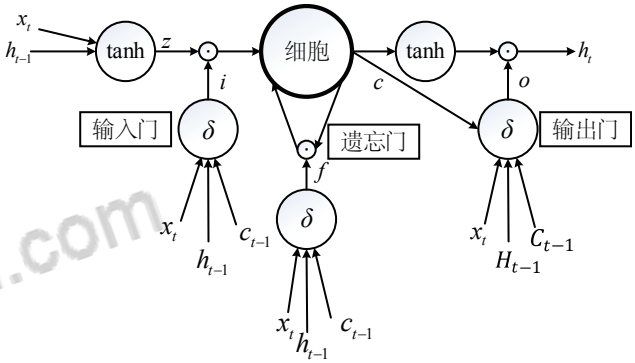


图 3 LSTM 网络细胞状态

图 3 中，当新输入一个 x_t 传入 LSTM 细胞节点，并与当前数据状态 h_{t-1} 汇合后，LSTM 细胞节点会按照如下顺序处理并训练自身。

1）遗忘门选择要从细胞状态 c_{t-1} 中丢弃的信息，并将其丢弃。

2）输入门解析输入数据，决定哪些信息需要存储在细胞状态中。

3）遗忘部分旧数据的细胞节点中被注入新数据后，输出门输出数据状态 h_t。

4）保存新的细胞状态 c_t 与当前数据状态 h_t，并将当前数据状态复制一份置入隐层。

以上即为 LSTM 细胞节点在一次输入后的行为，在实际应用中，输入门、输出门及遗忘门均有多个。图 3 中的各项参数定义如公式（2）到公式（7）所示。

$$i_t = \delta\left(W_{xi}x_t + W_{hi}H_{t-1} + W_{ci}C_{t-1} + b_i\right) \quad (2)$$

$$f_t = \delta\left(W_{xf}x_t + W_{hf}H_{t-1} + W_{cf}C_{t-1} + b_f\right) \quad (3)$$

$$z_t = \tanh\left(W_{xc}x_t + W_{hc}H_{t-1} + b_c\right) \quad (4)$$

http://www.xueshu.com

$$o_t = \delta\left(W_{xo}x_t + W_{ho}H_{t-1} + W_{co}C_t + b_o\right) \quad (5)$$

$$C_t = f_tC_{t-1} + i_tz_t \quad (6)$$

$$H_t = o_t\tanh\left(C_t\right) \quad (7)$$

其中，i, f, o 分别代表输入门、遗忘门与输出门处理后的结果，b 为在处理过程中的偏移量，C 代表细胞状态，W 为内部参数，δ 为 *sigmoid* 函数。最后得到的 H_t 为更新后输出的数据。

1.3 模型训练

模型训练的过程就是一个特征学习、去参数寻优的过程，在训练过程中参数 W 和 b 会不断地被更新，直至达到稳定值。由于训练集样本均带有漏洞利用代码，故在检查生成样本时可将满足 html 格式规范的样本称为合格样本，若一段时间内生成的总生成样本中 80% 以上都为合格样本，且此时模型的损失函数已经没有明显变化（每次迭代下降数值小于 0·1）时，判定模型已经收敛，则可以保存模型并用于样本生成。

选择损失函数为交叉熵，其定义如公式（8）所示。

$$Loss = -\sum_{t=1}^{T}\sum_{c=1}^{C} N_{t,c}\ln\hat{N}_{t,c} \quad (8)$$

其中，$N_{t,c}$ 代表第 t 个标签为标签库中第 c 种标签的实际概率值，$\hat{N}_{t,c}$ 代表第 t 个标签与标签库中第 c 种标签匹配的期望概率值。

为了使参数更新效率更高，本文使用的优化算法为 *rmsprop* 算法。

1.4 解码与样本生成

使用训练完成的模型输出数据，其数据格式为向量，为了得到完整的 html 文件，还需要对向量进行解析，输出向量分为标签向量和文本向量。标签向量由两个子向量组成，分别带有标签自身和标签属性的信息；文本向量也由两个子向量组成，分别带有文本自身和文本所在上一层标签的信息。

在空文件中插入标签与文本生成样本，其生成过程如下：

1）解析标签或文本向量，将其代表标签、标签属性、文本的数据与位置、频率信息分离。

2）将分离后的信息解码，得到标签、标签属性与文本数据的向量值。将其与样本解析时构成的字典进行比对，选取与字典中最相似 one-hot 值对应的标签、标签属性或文本数据作为向量解码得到的标签或文本。

3）将解码后的信息与对应的位置、频率值重新组装，获得解码后的向量 $\boldsymbol{N}_{ti}$、$\boldsymbol{N}_{ci}$。

4）选定一个维度 L_t，这个维度的值从 1 开始，代表着这个向量应该填充元素所在的位置。若处理对象为标签词向量，将其标签插入空文本中，再对其绑定的标签属性词向量进行同样的操作；若处理对象为文本词向量，则将文本插入位置为 L_t 的标签内。

5）重复步骤 4），直至输出向量处理完毕。

样本生成流程如图 4 所示，图 4 中使用 L_i 标记新文本中的位置 i，假设输出 3 个标签向量和一个文本向量。当标签向量中的标签属性词向量有多个选项且出现概率相近时，参考向量化过程中生成的标签向量集与文本向量集中的数据与样本预处理中生成的字典，在概率最高的 3 个属性中随机选择一个作为标签属性。

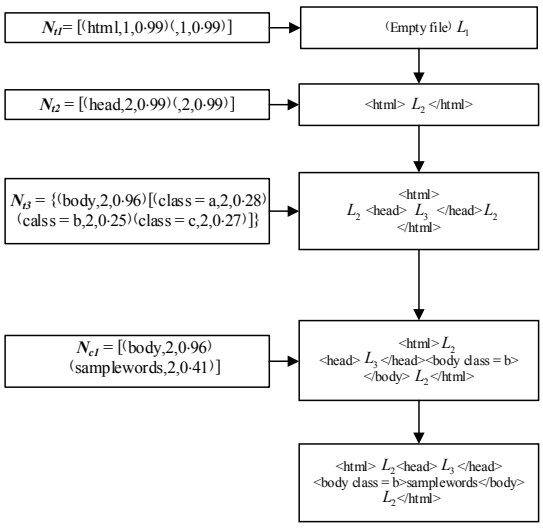


图 4 样本生成流程

为了在满足 html 文件格式的情况下使生成的样本具有更高的模糊测试价值，在生成过程中借鉴常用 Fuzz 工具生成样本时的变异策略 [15, 16] 对标签、标签

http://www.ixueshu.com

属性和文本进行一定程度的随机化变异操作，包括随机增加、删减、替换或更改标签对与标签属性值等 [17]。

2 实验过程与实验结论

2.1 实验过程

首先对源样本进行解析，从源样本中采集元素与标签的相关信息，并存入相应的元素库。神经网络根据元素库学习源样本的组成模式，学习完成后再生成测试样本。由于输入的源样本均为能产生 Crash 的恶意样本，由其作为学习材料生成的测试样本会带有源样本中的部分数据，故若测试样本符合 html 格式，则可以认为其具有 Fuzz 的价值。对生成的测试样本进行检查，若符合 html 格式的样本占 75% 以上，且神经网络损失函数数值在每次迭代时下降速率小于 0·1，则表示神经网络已经训练完成，可以用于样本生成。使用训练完成的神经网络生成测试样本，过滤掉不符合输入格式的样本，对剩下的样本进行模糊测试并监控测试结果、记录测试数据。

1）实验数据

本文选取 html 作为深度学习与 Fuzz 测试的数据类型，样本来源为 exploit-db 与 seebug 等漏洞库。由于此类样本数量较少，在训练前对样本集进行了扩充处理。对触发常见漏洞和使用常用攻击方式的样本进行了变异，增加该类样本所占的比重。最后得到测试集样本数量为 1400 个。

在选择训练集样本时，使用已经被证明能够造成浏览器崩溃或触发其他安全漏洞的样本作为原始训练集样本，根据样本利用的漏洞种类，如申请过大内存空间、释放后重用等将原始训练集分类，按照类型与原始样本数量使用类似于原始样本集的文件扩充训练集。扩充使用的文件选择基于该类漏洞的典型利用样本轻微变异的文件。由于无论是原始训练集样本还是扩充使用的样本文件均为能使浏览器功能异常，以此为训练集，经过神经网络处理后生成的样本带有部分针对浏览器漏洞的恶意代码，若生成的文件符合 html 格式，则认为其具有挖掘浏览器漏洞的价值。

2）实验环境与超参数

实验环境 ubuntu14·04，深度学习与模糊测试均在该平台下完成。深度学习框架选择谷歌研发的 TensorFlow。神经网络选用 LSTM 单向三层结构，超参数设置如表 2 所示。

表 2 超参数设置

超参数名	参数设置
LSTM 网络细胞数量	256
激活函数	*sigmoid*
Dropout 比例	0·3

使用 Morph 作为 Fuzz 测试工具，并引入异常检测模块 [18]，以有效样本数量为对比指标进行测试。Morph 提供了崩溃现场保存、当前文件记录等功能，可用于样本测试和崩溃追踪检测。模糊测试中使用 AFL（American Fuzz loop）中的插桩检测模块作为统计程序处理样本时的代码路径覆盖率。

3）实验分析

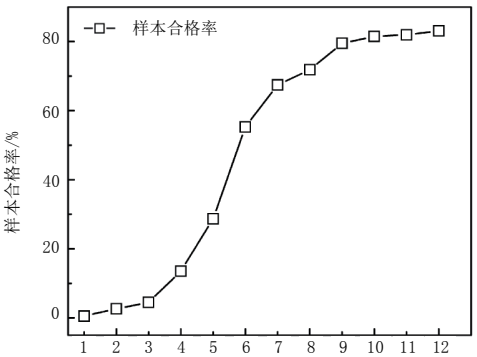
本文实验旨在测试由深度学习网络生成的 Fuzz 样本集的有效性与敏感性，将满足 html 文件格式规范的样本称为合格样本，合格样本在所有生成样本中的比例为样本合格率。将能产生 Crash 的样本称为敏感样本。以样本合格率、Fuzz 时间作为有效性验证指标，产生 Crash 的数量作为验证指标。

使用随机梯度下降的方式更新损失函数。输入所有样本，batch 值为 256，即每次迭代使用 256 个样本，直到遍历完所有输入样本，记录其损失函数变化趋势，统计最近 1 小时内生成样本的合格率。

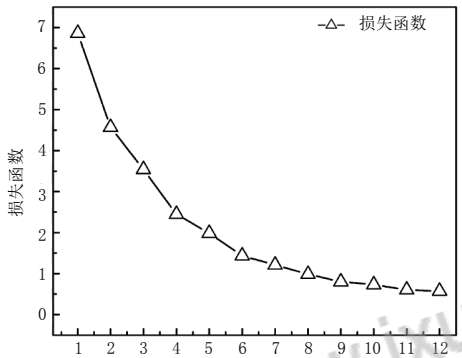
如图 5 a）所示，经过数十个小时的训练后，样本合格率达到 80%。如图 5 b）所示，神经网络的损失函数已经趋于稳定，基本满足实验需求。保存此时的模型，并将其用于样本生成。

使用模型生成 40000 个测试样本作为 Fuzz 测试的测试集，选定 Fuzz 对象为 linux 平台下的 midori 浏览器，为提升 Fuzz 效率，将测试集划分为等量的四份，开启四个线程同步 Fuzz，分别对每一批样本集

http://www.ixueshu.com



a)样本合格率



b)损失函数

图 5 神经网络学习情况

的 Fuzz 情况数据进行统计。统计数据包括样本数量、合格样本数量、Fuzz 时间、产生 Crash 的数量等，同时使用 AFL 统计代码覆盖率。Fuzz 样本数据与 Fuzz 测试结果统计数据分别如表 3 所示。

表 3 Fuzz 样本数据

样本批次	样本数量 / 个	合格样本数量 / 个	样本合格率 /%
第一批	10000	8312	83·12
第二批	10000	8001	80·01
第三批	10000	8362	83·62
第四批	10000	8196	81·96

以 LSTM 为基础设计的深度神经网络在经过学习后，生成样本合格率在 80% 左右波动。通过对 Fuzz 结果的统计与分析，发现生成样本质量已经趋于稳定。各批次样本的 Fuzz 时间、Crash 数量、路径覆盖率等参数值仅小幅波动。

将产生的 Crash 按照类型划分，并统计其触发次数。对敏感样本逐个分析，检测其结构、内容与源样本的一致性，若代码重复率达到 80%，将其标记为衍生样本，表示该样本与学习集中某些样本为亲属关系。统计结果如表 4 所示。在 Fuzz 过程中，发现了上百个 Crash，而且大部分 Crash 可以判断发生原因，说明该模型在实验环境下能作为浏览器 Fuzz 的样本集生成框架。

表 4 敏感样本集分析结果

Crash 类型	触发次数	衍生样本数量	非衍生样本数量
内存破坏	31	25	6
网页过大	16	12	4
其他	36	23	13

在未来的训练中，将使用非衍生样本作为训练集的扩充，以扩大测试面积与代码覆盖率。

2.2 实验结论

通过对实验结果的统计与分析可以得出实验结论，在样本生成方面，经过一段时间的学习后，LSTM 神经网络能够批量生成符合 html 文件结构的样本，通过对样本进行随机变异保证了样本多样性，生成样本的合格率基本满足实际应用的要求。在模糊测试方面，在 Fuzz 方向较少的情况下，路径覆盖率依旧在 55% 以上，能覆盖目标程序大部分代码，并能产生 Crash，大部分敏感样本都针对三类典型漏洞。神经网络生成的样本集中存在新的有效畸形样本，体现出神经网络的学习能力，在更多次学习与训练后，挖掘效率有望进一步提高。

3 结束语

深度学习作为研究热点，在网络安全方面的应用发展迅猛，相较于传统的 Fuzz 样本生成策略而言，深度学习指导二进制程序输入集生成更有针对性。然而，目前针对 html 文件进行解析与生成的工作仍未能大批量投入工程中使用。针对浏览器 Fuzz 中没有针对性的随机化生成样本、重复样本多、对敏感点没有侧重 Fuzz 的问题，本文提出了相应的解决方案，实现了 html 样本生成，使用深度学习生成的样本集来进行 Fuzz 测试，并取得了一定成果，证实了将深度学习应用到浏览器 Fuzz 上是可行的。未来将会进一步

http://www.ixueshu.com

研究深度学习工具与理论，提高生成样本合格率与针对性，从而进一步提高样本集生成效率与挖掘漏洞的可能性。同时，根据样本生成情况和 Fuzz 结果进行反馈，使用强化学习指导变异样本生成也是可行的研究方向。（责编 程斌）

参考文献：

[1] WANG xiajing, HU changzhen, MA rui. A Survey of Key techniques of Binary Program Vulnerability Discovery[J]. Netinfo Security, 2017, 17(8): 1–13.
王夏菁,胡昌振,马锐,等. 二进制程序漏洞挖掘关键技术研究综述 [J]. 信息网络安全, 2017, 17(8): 1–13.

[2] PANG Y, XUE X, NAMIN A S. Early Identification of Vulnerable Software Components via Ensemble Learning[C]// IEEE. 15th IEEE International Conference on Machine Learning and Applications(ICMLA), December 18–20, 2016, Los Angeles, California, USA. New York: IEEE, 2017: 476–481.

[3] PANG Y, XUE X, NAMIN A S. Predicting Vulnerable Software Components through N–Gram Analysis and Statistical Feature Selection[C]// IEEE. 14th International Conference on Machine Learning and Applications (ICMLA), December 4 –7, 2013, Miami, Florida, USA. New York: IEEE, 2015:543–548.

[4] WEN weiping, WU bozhi, JIAO yingnan, et al. Design and Implementation on Malicious Document Detection Tool Based on Machine Learning [J]. Netinfo Security, 2018, 18 (8): 1–7.
文伟平, 吴勃志, 焦英楠, 等. 基于机器学习的恶意文档识别工具设计与实现 [J]. 信息网络安全, 2018, 18(8): 1–7.

[5] GODEFROID P, PELEG H, SINGH R. Learn&Fuzz: Machine Learning for Input Fuzzing[C]// IEEE. 32nd IEEE/ACM International Conference on Automated Software Engineering, October 30–November 3, 2017, Urbana–Champaign, IL, USA, New York: IEEE, 2017: 50–59.

[6] WANG J, CHEN B, WEI L, et al. Skyfire: Data–Driven Seed Generation for Fuzzing[C]//IEEE. 38th IEEE Symposium on Security and Privacy, May 22–24, 2017, San Jose, California, USA. New York: IEEE, 2017: 579–594.

[7] WU Fang, A Study of Binary Vulnerability Analysis and Detection Based on Deep Leraning[D]. Beijing: Beijing Jiaotong University, 2018.
吴芳. 基于深度学习的二进制程序漏洞分析与检测方法研究 [D]. 北京: 北京交通大学, 2018.

[8] SUNDERMEYER M, SCHLÜTER R, NEY H. LSTM Neural Networks for Language Modeling[C]//INTERSPEECH. 13th Annual Conference of the International Speech Communication Association, September 9–13, 2012, Portland, Oregon, USA. New York: 2012: 601–608.

[9] KALCHBRENNER N, GREFENSTETTE E, BLUNSOM P. A Convolutional Neural Network for Modelling Sentences[EB/OL]. https://arxiv.org/abs/1404.2188,2014–4–8.

[10] PALANGI H, DENG L, SHEN Y, et al. Deep Sentence Embedding Using Long Short–Term Memory Networks: Analysis and Application to Information Retrieval[J]. ACM Transactions on Audio Speech & Language Processing, 2016, 24(4): 694–707.

[11] BENGIO Y, SIMARD P, FRASCONI P. Learning Long–term Dependencies with Gradient Descent is Difficult[J]. IEEE Transactions on Neural Networks, 2002, 5(2):157–166.

[12] BELTRAMELLI T. Pix2code: Generating Code from a Graphical User Interface Screenshot[C]//EICS. ACM SIGCHI Symposium on Engineering Interactive Computing Systems, June 26–29, 2017, Lisbon, Portugal. New York: ACM, 2018: 3.

[13] NIEPERT M, AHMED M, KUTZKOV K. Learning Convolutional Neural Networks for Graphs[C]//ICML. 33rd International conference on machine learning, June 19 – 24, 2016, New York City, NY, USA. New York: ICML, 2016: 2014–2023.

[14] GRAVES A. Generating Sequences With Recurrent Neural Networks[EB/OL]. https://arxiv.org/abs/1308.0850, 2013–8–4.

[15] SHE D, PEI K, EPSTEIN D, et al. NEUZZ: Efficient Fuzzing with Neural Program Learning[EB/OL]. https://arxiv.org/abs/1807.05620, 2018–7–15.

[16] RAJPAL M, BLUM W, SINGH R. Not all Bytes are Equal: Neural Byte Sieve for Fuzzing[EB/OL]. https://arxiv.org/abs/1711.04596, 2017–11–10.

[17] BÖTTINGER K, GODEFROID P, SINGH R. Deep Reinforcement Fuzzing[C]//IEEE. IEEE Security and Privacy Workshops (SPW), May 24, 2018, San Francisco, CA, USA. New York: IEEE, 2018: 116–122.

[18] HUANG Yi. Research on Software Security Vulnerability Discovery Based on Fuzzing[D]. Hefei: University of Science and Technology of China, 2010.
黄奕. 基于模糊测试的软件安全漏洞发掘技术研究 [D]. 合肥: 中国科学技术大学, 2010.

http://www.ixueshu.com

免费论文查重：http://www.paperyy.com

3亿免费文献下载：http://www.ixueshu.com

超值论文自动降重：http://www.paperyy.com/reduce_repetition

PPT免费模版下载：http://ppt.ixueshu.com