Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

基于深度学习的二进制软件漏洞挖掘

动态缺陷检测方法

硕士研究生　闫晗

2020年8月2日

内容提要

- 背景简介
- 基本概念
- 算法原理
- 优劣分析
- 应用总结
- 参考文献

背景简介

- 预期收获
  - 1. 了解二进制缺陷检测任务的基本概念和TIPO
  - 2. 了解模糊测试、符号执行等动态缺陷检测方法
  - 3. 理解基于深度学习的动态二进制缺陷检测方法

基本概念

基本概念

- 漏洞
  - 软件漏洞 (Vulnerablity) 是由软件在设计、开发或配置过程中存在的错误 (Mistake) 而导致的缺陷 (Flaw) 的实例，利用漏洞可以违反某些显式或隐式的安全策略。


```
//STACK_OVER_FLOW
void stack_over_flow(unsigned char* data)
{
    //缓冲区
    unsigned char buffer[BUF_LEN];

    //拷贝数据至缓冲区
    strcpy((char*)buffer, (char*)data);
}
```

错误：拷贝动作没有约束边界

缺陷：可能导致栈缓冲区溢出

漏洞：可能导致恶意代码执行

基本概念

- 漏洞挖掘

T	发现目标软件中存在的漏洞
I	目标软件（源程序或二进制程序）
P	1. 软件缺陷检测（缺陷位置）； 2. 可利用性分析（利用方法）
O	目标软件的漏洞信息（缺陷位置、利用方法等）

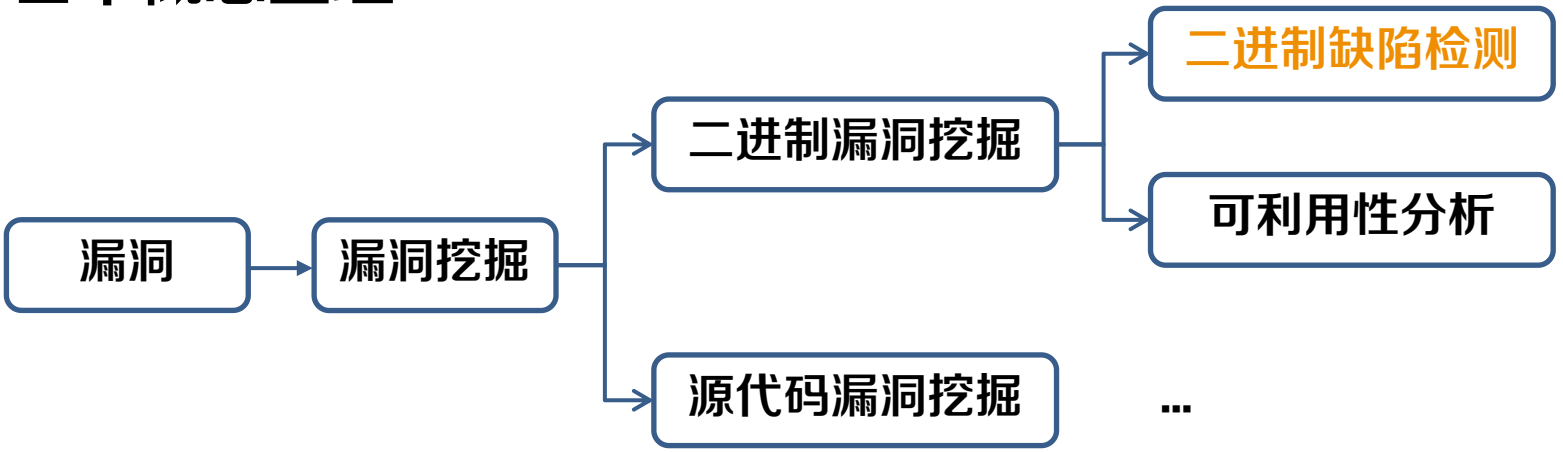
- 二进制漏洞挖掘
  - 目标软件是二进制文件的漏洞挖掘
  - 二进制文件包括可执行文件、链接库文件等

基本概念

- 二进制缺陷检测

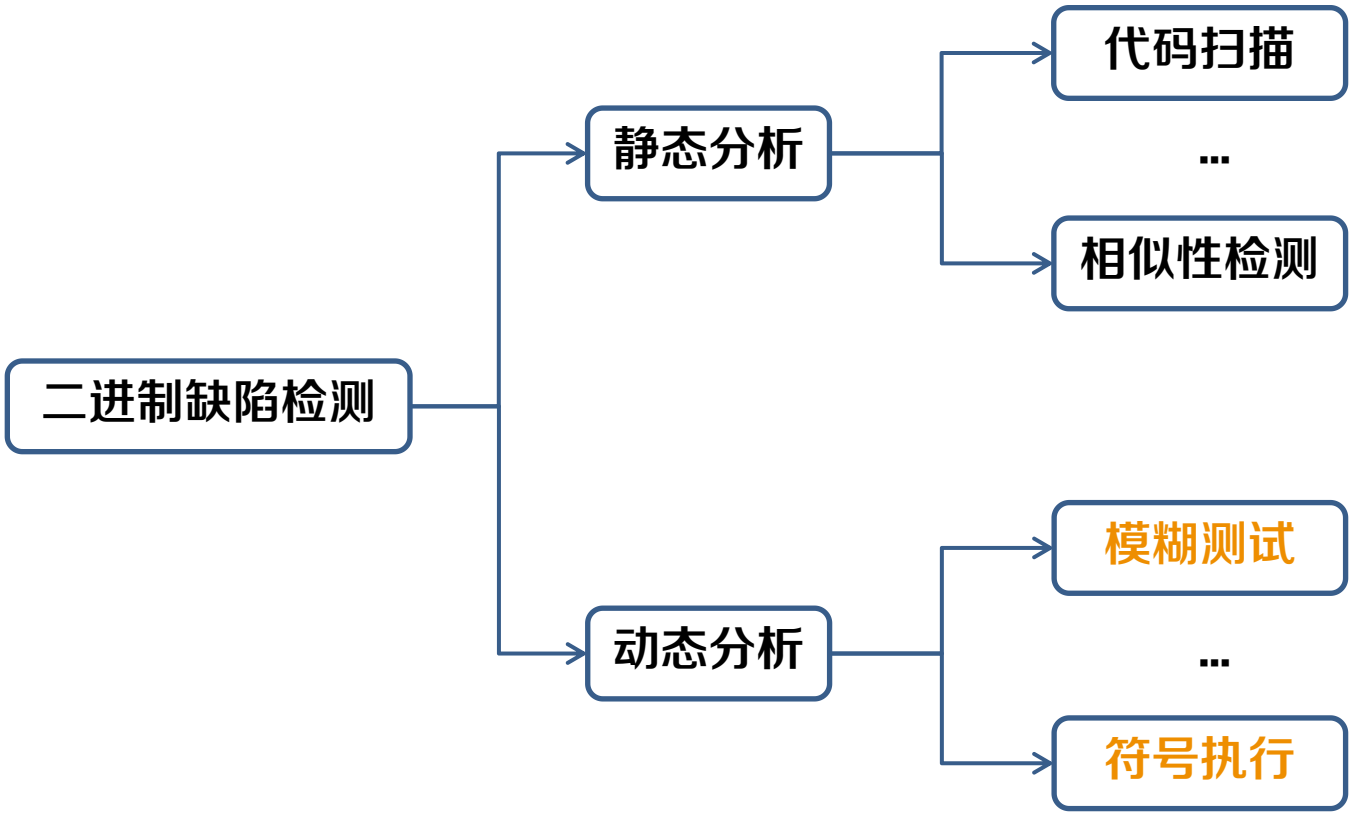
T	发现并定位二进制程序中存在的缺陷代码
I	目标二进制程序
P	静态分析/动态分析
O	目标二进制程序的缺陷代码位置

- 基本概念整理



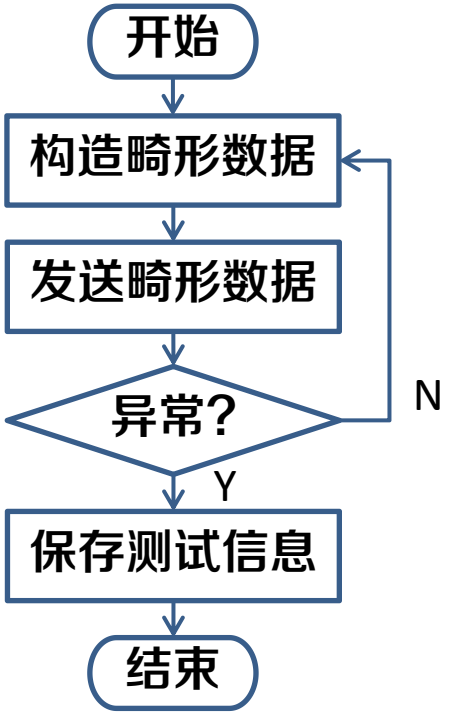
基本概念

- 二进制缺陷检测
  - 主要技术和方法



基本概念

- 二进制缺陷检测（模糊测试）
  - 基本概念：通过构造随机的、非预期的畸形数据作为程序的输入，并监控程序执行过程中可能产生的异常，之后将这些异常作为分析的起点，确定其可利用性。
  - 算法流程
    ① 构造非预期的畸形数据
    ② 向目标程序发送畸形数据
    ③ 目标程序产生异常则记录
    ④ 未发现异常则跳转到 ①
    ⑤ 对测试结果进行分析



基本概念

- 二进制缺陷检测（模糊测试）
  - 基于变异的模糊测试
    - 经典方法：随机变异（例如AFL工具）

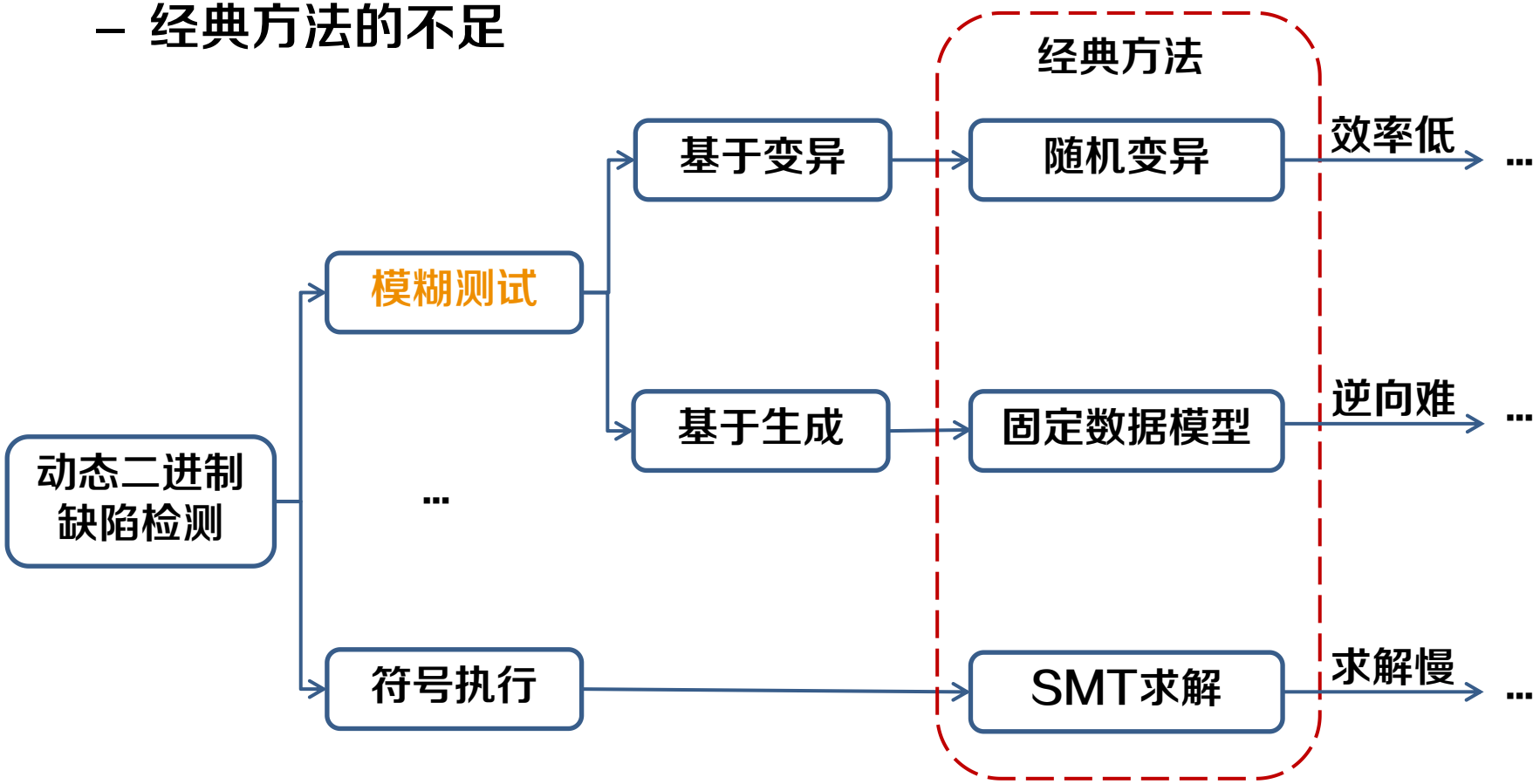
  - 基于生成的模糊测试
    - 经典方法：基于固定数据模型生成（例如Peach工具）

基本概念

- 二进制缺陷检测（符号执行）
  - 基本概念：符号执行是一种能够系统性探索程序执行路径的程序分析技术，通过分析程序求解可以使特定代码区域执行的输入。使用符号执行分析程序时，程序的输入为符号值。
  - 算法优势：通过符号执行方法产生的测试输入与执行路径之间具有一对一的关系，能够避免冗余测试输入的产生，进而有效解决模糊测试冗余测试用例过多导致的代码覆盖率增长慢的问题。

基本概念

- 二进制缺陷检测
  - 经典方法的不足



算法原理

算法原理

- 基于深度学习的模糊测试（基于变异）
  - TIPO

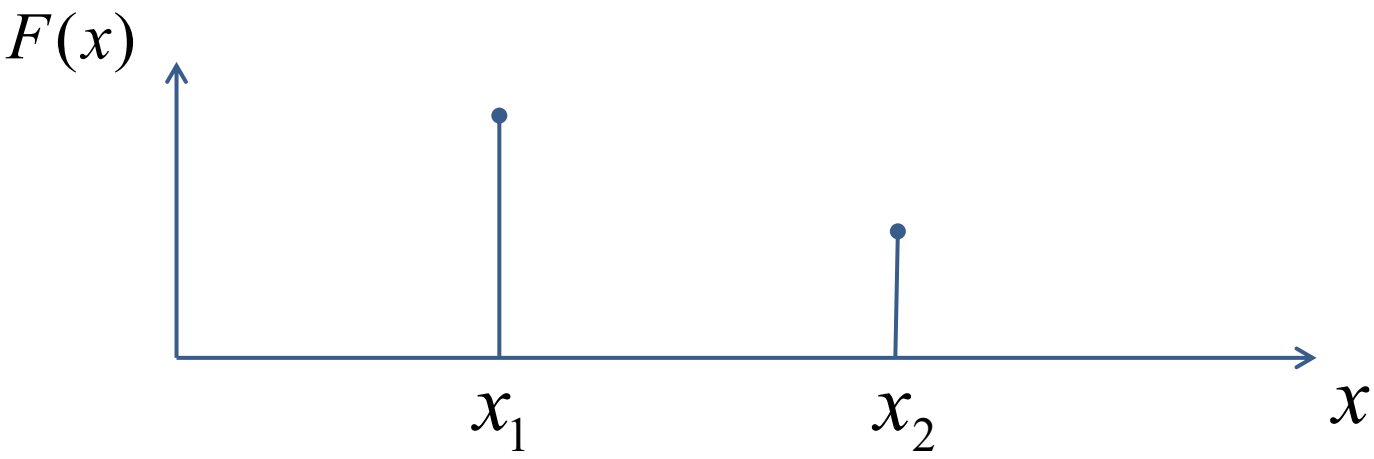
T	优化样本变异策略，提升模糊测试效率
I	目标程序
P	1. 训练神经网络，近似程序分支行为 2. 梯度优化输入样本，识别出最优位置进行变异
O	变异样本

P	程序分支行为不连续，无法直接进行梯度优化
C	输入长度固定
D	平滑不连续的程序分支行为
L	IEEE Symposium on Security & Privacy

算法原理

- 基于深度学习的模糊测试（基于变异）
  - 优化问题
    - x： 程序输入 $x \in X$
    - $F(x)$：x 触发的异常数
    - $C(x)$： 生成的 x 的集合

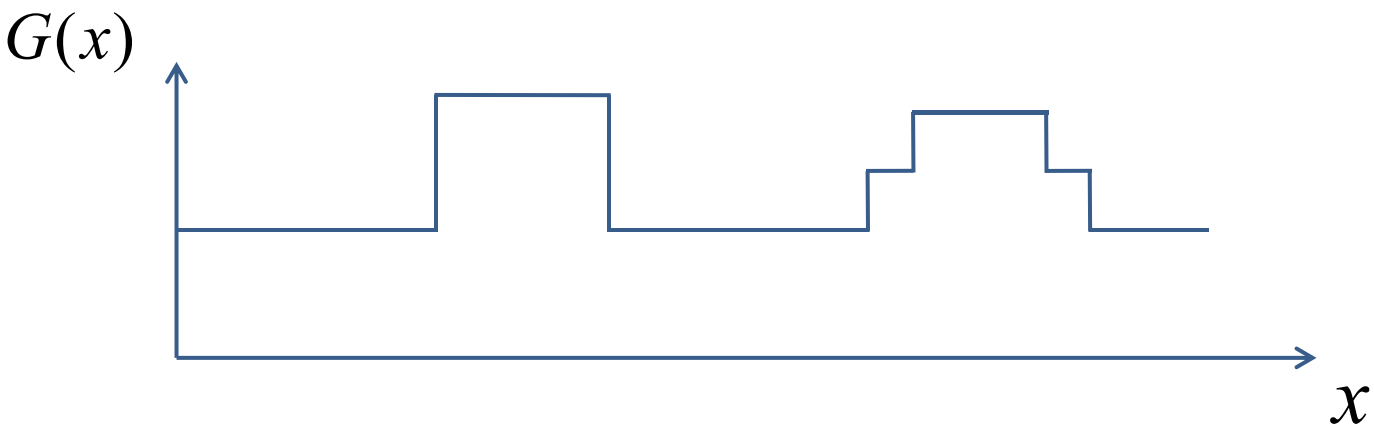
$$Maximize \sum_{x \in C(X)} F(x)$$



算法原理

- 基于深度学习的模糊测试（基于变异）
  - 优化问题
    - x：程序输入 $x \in X$
    - $G(x)$：x 覆盖的（新的）控制流边数
    - $C(x)$：生成的 x 的集合

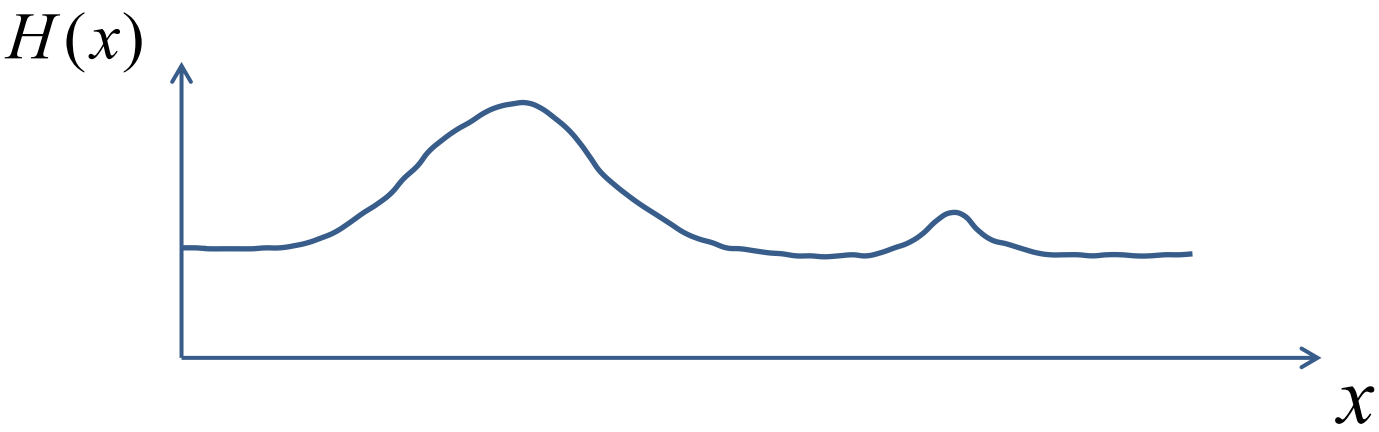
$$Maximize \sum_{x \in C(X)} G(x)$$



算法原理

- 基于深度学习的模糊测试（基于变异）
  - 优化问题
    - x：程序输入 $x \in X$
    - $H(x)$：x 覆盖的（新的）控制流边数的平滑近似
    - $C(x)$：生成的 x 的集合

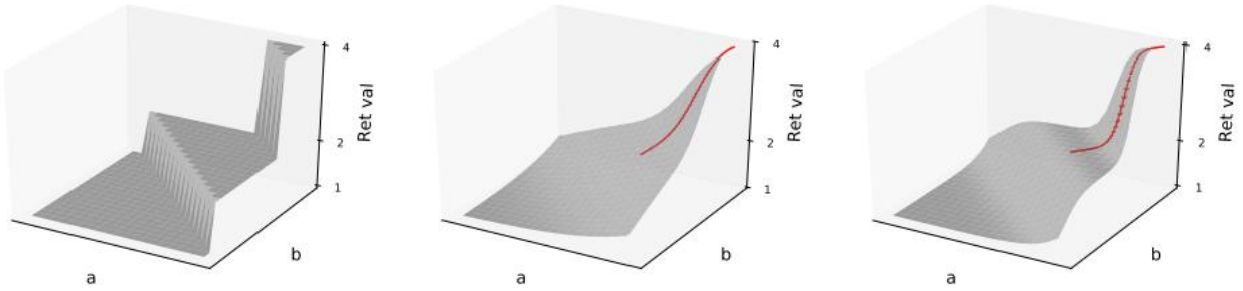
$$Maximize \sum_{x \in C(X)} H(x)$$



算法原理

- 基于深度学习的模糊测试（基于变异）
  - 实例分析

```
1  z = pow(3, a+b);
2  if(z < 1){
3    return 1;
4  }
5  else if(z < 2){
6    return 2;
7  }
8  else if(z < 4){
9    //vulnerability
10   return 4;
11 }
```

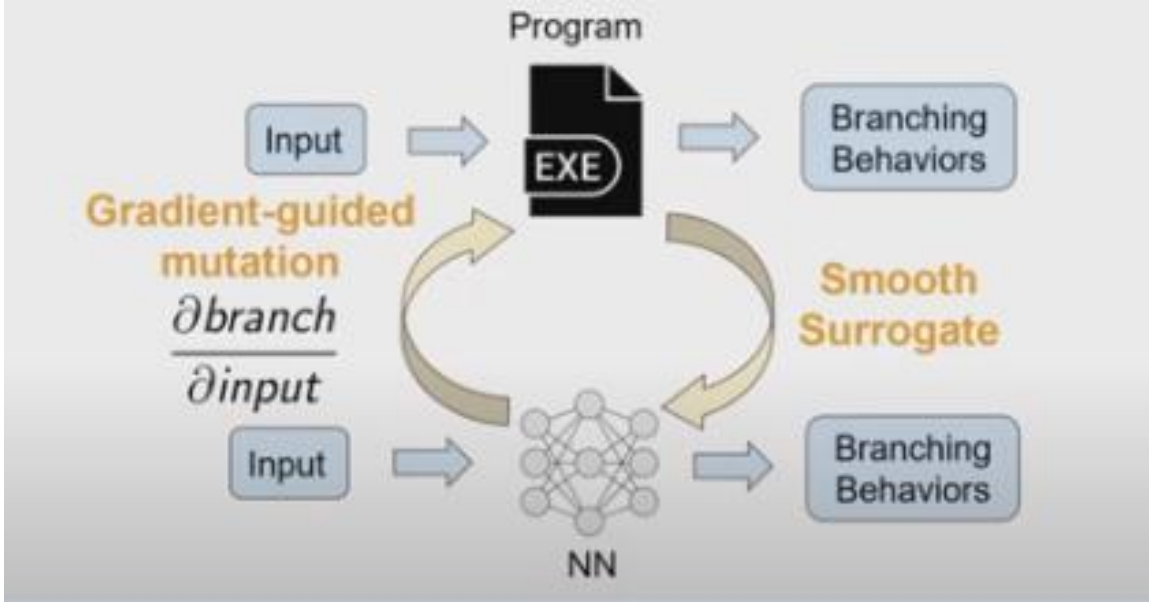


(a) Original (b) NN smoothing (c) NN smoothing + refining

  - 关键技术
    - 平滑近似：找到模拟程序分支行为的连续函数
    - 梯度优化：识别出最优的样本变异位置

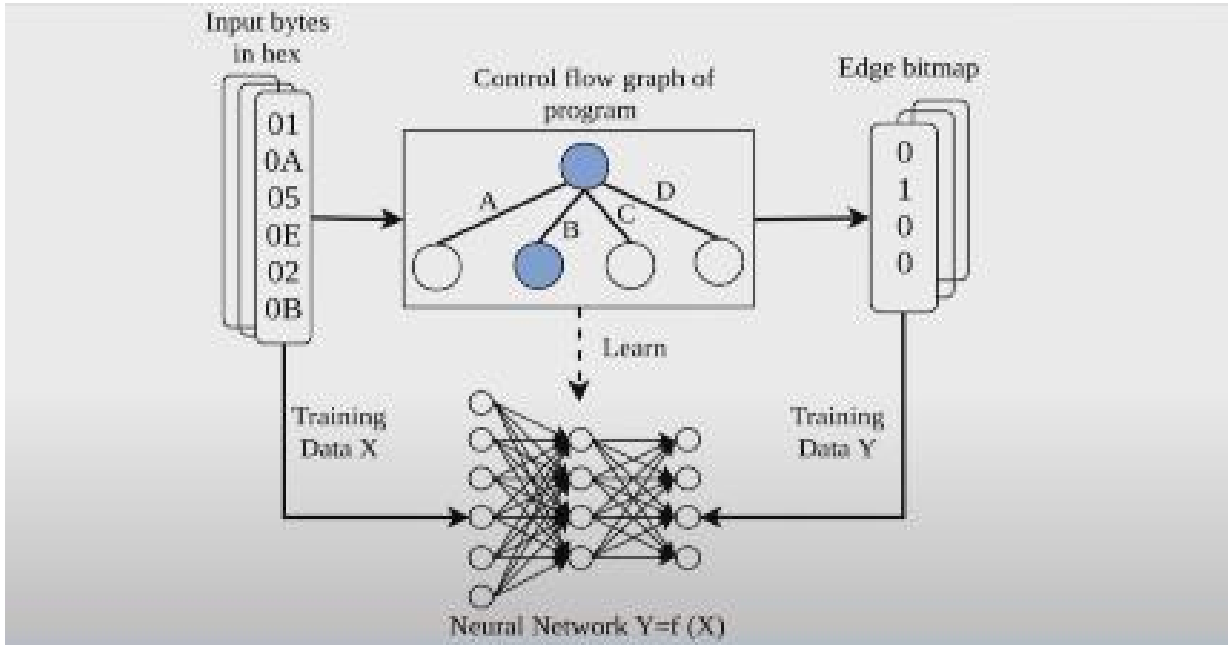
算法原理

- 基于深度学习的模糊测试（基于变异）
  - 平滑近似
    - 利用神经网络模拟程序的分支行为
    - 对于相同的输入，产生相同的输出



算法原理

- 基于深度学习的模糊测试（基于变异）
  - 平滑近似
    - 数据：输入数据（十六进制字节序列）
    - 标签：覆盖边的bitmap（通过AFL获取）



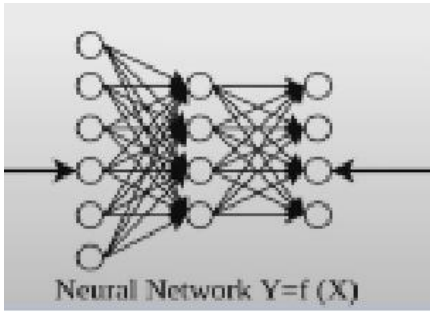
算法原理

- 基于深度学习的模糊测试（基于变异）
  - 梯度优化

Algorithm 1 Gradient-guided mutation

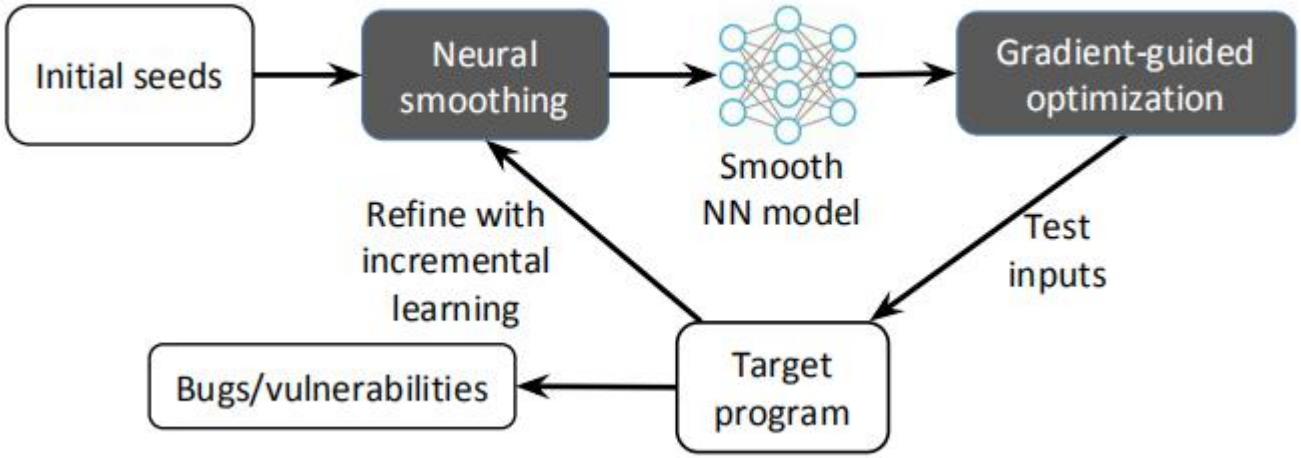
Input: $seed \leftarrow$ initial seed
$iter \leftarrow$ number of iterations
$k \leftarrow$ parameter for picking top-k critical bytes for mutation
$g \leftarrow$ computed gradient of seed

1: **for** $i = 1$ to $iter$ **do**
2: $locations \leftarrow top(g, k_i)$
3: **for** $m = 1$ to 255 **do**
4: **for** $loc \in locations$ **do**
5: $v \leftarrow seed[loc] + m * sign(g[loc])$
6: $v \leftarrow clip(v, 0, 255)$
7: $gen\_mutate(seed, loc, v)$
8: **for** $loc \in locations$ **do**
9: $v \leftarrow seed[loc] - m * sign(g[loc])$
10: $v \leftarrow clip(v, 0, 255)$
11: $gen\_mutate(seed, loc, v)$



算法原理

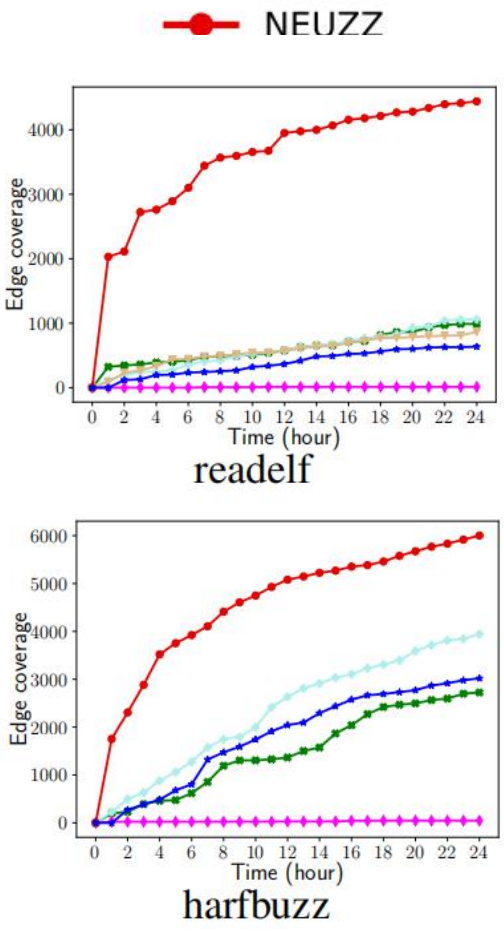
- 基于深度学习的模糊测试（基于变异）
  - 原理框图
    - 训练CNN网络学习连续可微的神经程序，用于近似模拟程序的实际执行逻辑，并通过梯度下降指导测试用例生成。
    - 优点：程序分支覆盖率高



算法原理

- 基于深度学习的模糊测试（基于变异）
  - 实验结果（LAVA-M）

Programs	AFL	AFLFast	VUzzer	KleeFL	AFL-laf-intel	NEUZZ
Detected Bugs per Project						
readelf	4	5	5	3	4	16
nm	8	7	0	0	6	9
objdump	6	6	0	3	7	8
size	4	4	0	3	2	6
strip	7	5	2	5	7	20
libjpeg	0	0	0	0	0	1
Detected Bugs per Type						
out-of-memory	✓	✓	✗	✓	✓	✓
memory leak	✓	✓	✓	✓	✓	✓
assertion crash	✗	✓	✗	✗	✓	✓
interger overflow	✗	✗	✗	✗	✗	✓
heap overflow	✓	✗	✗	✗	✗	✓
Total	29	27	7	14	26	60



算法原理

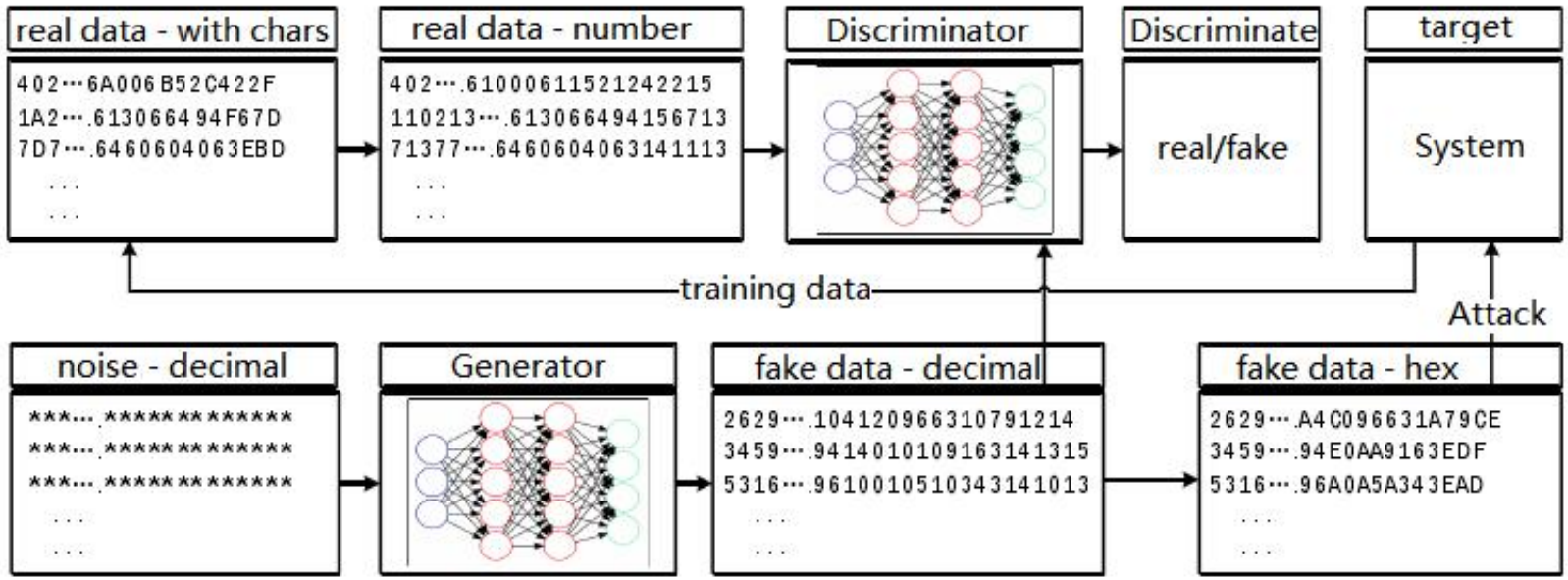
- 基于深度学习的模糊测试（基于生成）
  - TIPO

T	学习目标协议规则，生成模糊测试样本
I	真实协议数据
P	1. 训练生成器、判别器 2. 每10个训练周期保存生成模型
O	符合目标协议规则的测试样本

P	目标协议规则未知，逆向困难
C	可以获取真实协议数据
D	自动学习目标协议规则
L	CCF B类期刊

算法原理

- 基于深度学习的模糊测试（基于生成）
  - 原理框图
    - 通过生成器生成符合协议规则的样本

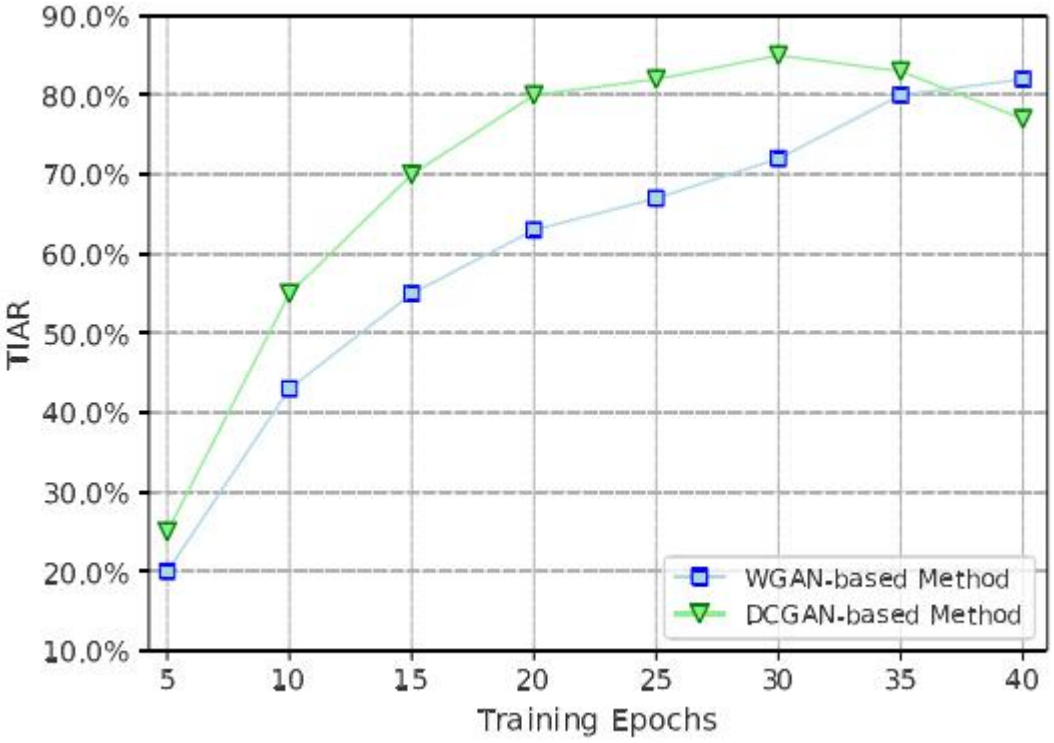


算法原理

- 基于深度学习的模糊测试（基于生成）
  - 模型训练
    - 协议漏洞挖掘任务的目标是尽可能多地引发系统异常行为，而非拟合协议数据分布
    - 通过保持生成数据与真实数据的差异程度可以提高模型发现漏洞的能力。
    - 在训练模型的过程中，每10个训练周期保存对应的生成模型。不同的生成模型因为训练程度的差别，其生成的模糊测试数据将与真实数据维持多个差异度，提升测试效果。

算法原理

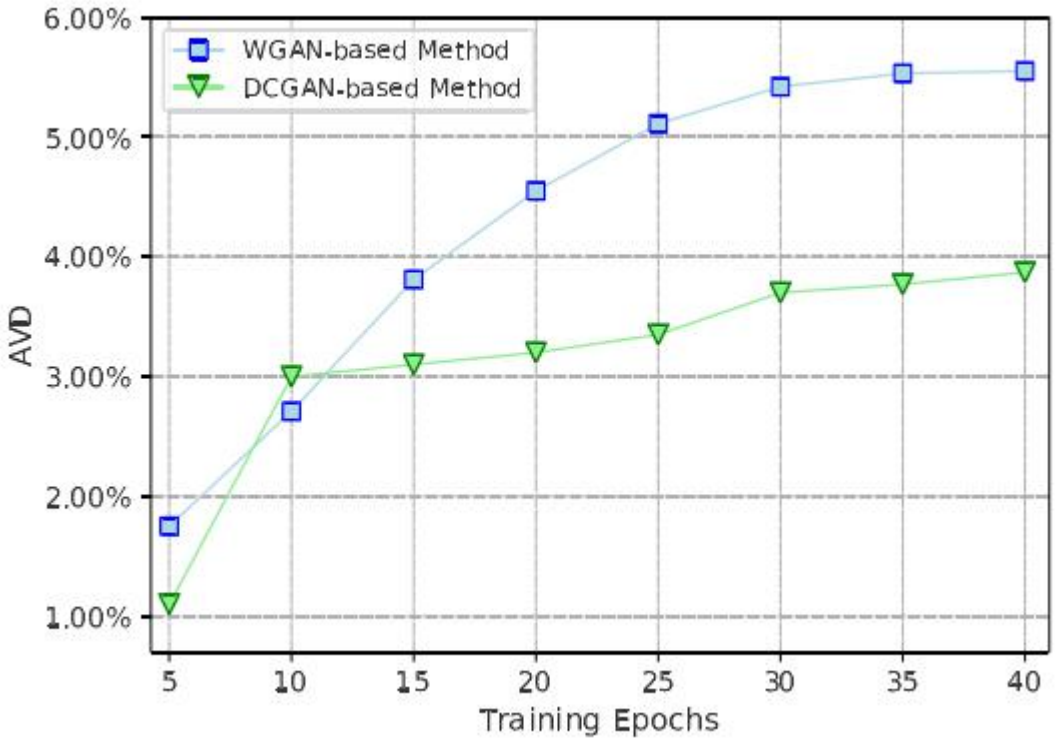
- 基于深度学习的模糊测试（基于生成）
  - 实验结果
    - TIAR（Test Input Accept Rate）测试用例接受率



$$TIAR = \frac{nAccept}{nSent} \times 100\%$$

算法原理

- 基于深度学习的模糊测试（基于生成）
  - 实验结果
    - AVD（Ability of Vulnerability Detection）



$$AVD = \frac{nBugs}{nCases} \times 100\%$$

算法原理

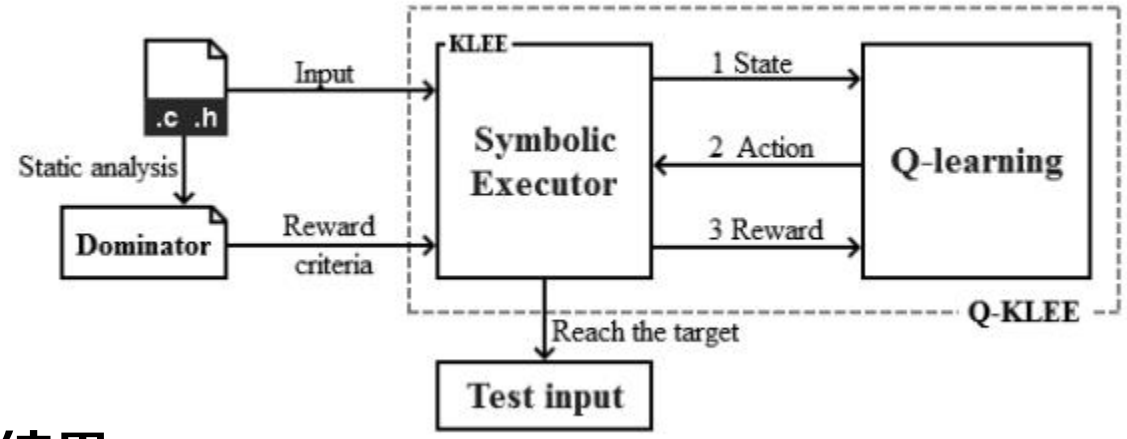
- 基于强化学习的符号执行
  - TIPO

T	生成可达指定语句的测试输入
I	指定待执行的语句
P	1. 如果执行至指定语句的支配节点，则更新Q表 2. 根据Q表选择执行路径
O	指定语句的测试输入

P	路径爆炸
C	白盒测试
D	对路径选择加以制导
L	CCF B类会议

算法原理

- 基于深度学习的符号执行
  - 原理框图



  - 实验结果

KLEE			Q-KLEE		
# Paths	# Instructions	# Time(s)	# Paths	# Instructions	# Time(s)
1,489	1,020,765	13.68	139	90,892	18.34

优劣分析

- 对比分析
  - 静态分析执行效率高、动态分析代码误报率低
  - 基于深度学习的漏洞挖掘方法自动化程度高
  - 基于深度学习的模糊测试和符号执行的效率高

应用总结

- 基于变异的模糊测试
  - 输入数据非结构化的场景
- 基于生成的模糊测试
  - 输入数据结构化的场景
- 符号执行
  - 为模糊测试提供制导

参考文献

[1] She D , Pei K , Epstein D , et al. NEUZZ: Efficient Fuzzing with Neural Program Smoothing[C]. 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019.

[2] Li Z , Zhao H , Shi J , et al. An Intelligent Fuzzing Data Generation Method Based on Deep Adversarial Learning[J]. IEEE Access, 2019, 7:49327-49340.

[3] Wu J , Zhang C , Pu G . Reinforcement Learning Guided Symbolic Execution[C]. 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2020.

谢谢！

大成若缺，其用不弊。大盈若冲，其用不穷。大直若屈。大巧若拙。大辩若讷。静胜躁，寒胜热。清静为天下正。