Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

软件灰盒定向模糊测试技术

博士研究生 张浩然

2025年06月29日

问题回溯

- 总结反思
  - 讲解语气过于平淡
  - 算法选取较为简单

- 相关内容
  - 2025.06.22 杨语航《大模型指导的内核模糊测试》
  - 2024.05.18 徐菊彬《大模型指导的协议模糊测试》
  - 2024.09.03 张浩然《大模型赋能的模糊测试用例生成技术》
  - 2024.06.09 谢宁　《基于变异的模糊测试》
  - 2024.05.26 邵思源《面向网络应用程序的模糊测试》

内容提要

- 预期收获
- 题目内涵解析
- 研究背景与意义
- 研究历史与现状
- 知识基础
- 算法原理
  - SELECTFUZZ
  - DeepGo
- 特点总结与工作展望
- 参考文献

预期收获

- 预期收获
  - 了解软件灰盒定向模糊测试的方法背景和基本概念
  - 理解定向模糊测试的通用思路和应用场景
  - 掌握深度学习、强化学习在定向模糊测试中的使用方式

内涵解析与研究目标 软件灰盒定向模糊测试技术

- 研究目标
  - 在软件功能日渐复杂、全面的背景下，保留灰盒模糊测试效率，引导测试更快速地覆盖特定目标路径或触发特定漏洞
- 题目内涵解析
  - 模糊测试(Fuzz)：通过向测试对象提供大量畸形测试用例作为输入，并监视其错误响应，以揭示潜在的异常缺陷及安全漏洞
  - 灰盒模糊测试：针对二进制测试对象，利用覆盖率反馈信息引导测试方向
  - 定向模糊测试：引导模糊测试朝特定目标快速逼近的技术

研究背景 软件灰盒定向模糊测试技术

- 研究背景
  - 模糊测试的兴起：模糊测试作为一种自动化测试技术，通过生成随机或半随机的输入数据，能够有效发现未知漏洞
  - 灰盒模糊测试：与白盒、黑盒测试相比，兼顾测试效率与测试方向选择
  - 通用模糊测试专注于触发尽可能多的代码块，在软件功能复杂、分支较多时，无法高效的测试易出现问题的区域
- 研究意义
  - 提升模糊测试对特定高风险目标的覆盖效率，增强漏洞发现的针对性与实用性

研究历史与现状 软件灰盒定向模糊测试技术

Böhme等人首次提出了针对灰盒模糊测试的AFLGo，通过生成有针对性的输入，提高对特定目标(如公开的CVE漏洞或特定的代码行位置等)测试的效率和覆盖率

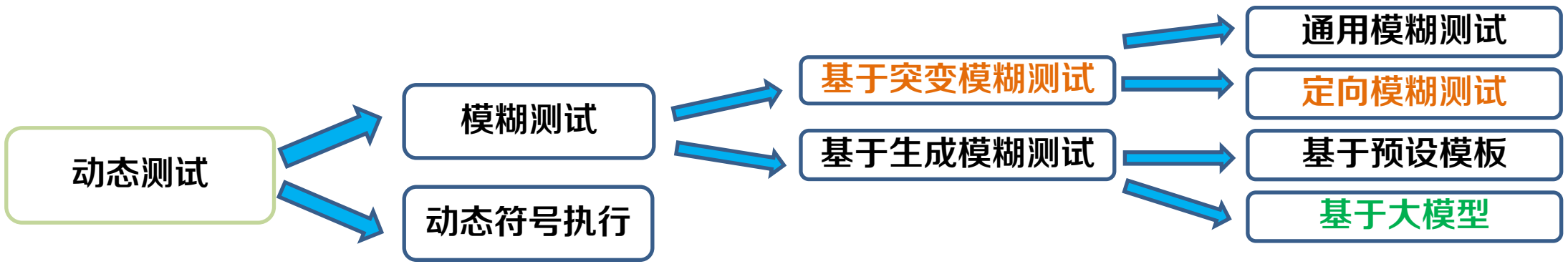
SelectFuzz进一步细化了剪枝方法，将可达路径划分为目标相关路径和不相关路径，对不相关路径进行细化修剪，同时引入了更精确的距离计算方法，优化测试资源分配

VULSEYE针对智能合约设计了一种基于状态的定向灰盒模糊器，它通过静态分析识别漏洞敏感的代码目标，利用逆向分析确定能够引发漏洞的合约状态目标，确保测试聚焦于可能导致漏洞的状态目标

2017 2022 2023 2024 2025

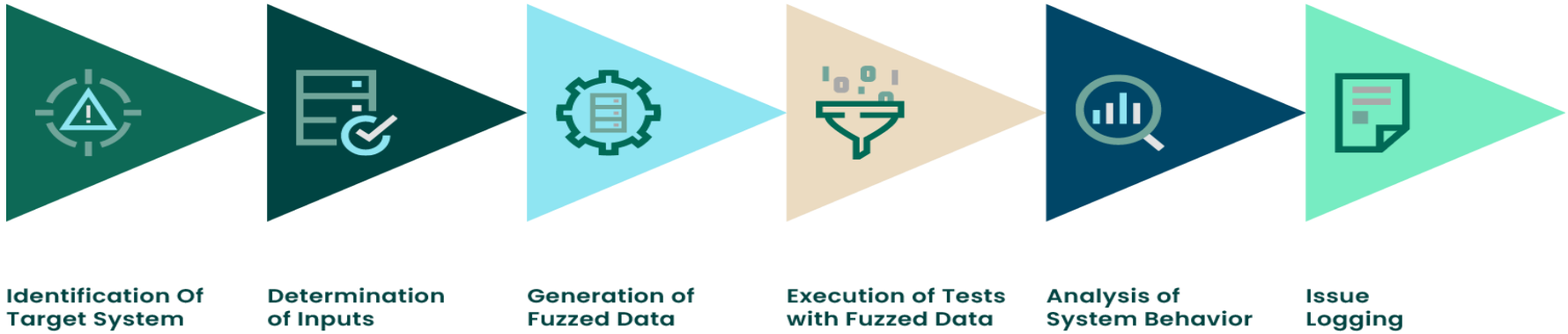
BEACON是首个在预处理阶段修剪不可达路径的方法，通过静态分析程序，删除无法到达目标代码的基本块，并通过反向区间分析进一步剔除更多不可达路径，专注于减少测试过程中不必要的路径探索

DeepGo通过结合历史路径信息和深度学习模型，利用神经网络模拟和预测路径转换过程，生成具有更高序列奖励的路径转换序列，提升模糊器触发目标的效率



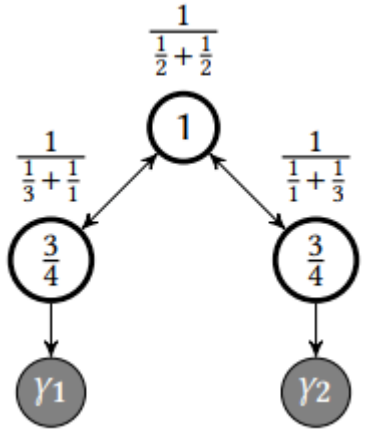
知识基础 定向模糊测试

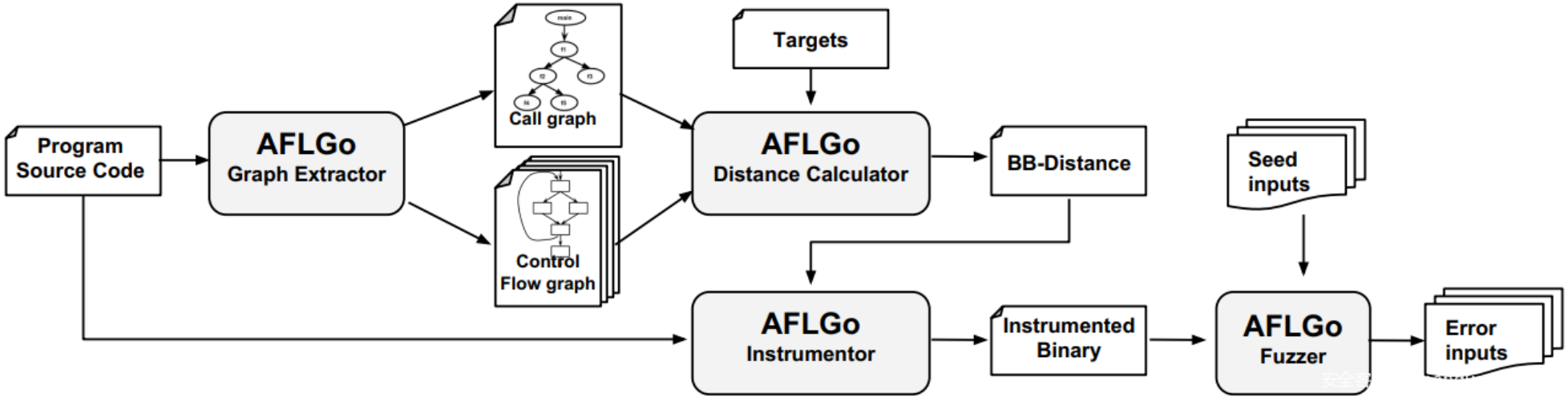
- 通用模糊测试
  - 生成大量随机输入数据，提供给目标程序运行，检测程序在异常输入下的行为
  - 高效检测目标程序中在任意位置存在的漏洞
  - 目标程序代码量大、功能复杂时效率低下
- 定向模糊测试(Directed Fuzzing) = 通用模糊测试 + 针对目标位置的引导机制
  - 在模糊测试的基础上，将更多测试资源放在目标特定位置
  - 指定最新patch修改的代码行，测试是否引入新的崩溃
    - 如gcc的-fstrict-overflow功能激进优化整数运算，但是可能有溢出问题
    - 指定目标特定代码行fold-const.c:line123，将测试资源重点集中于触发此处的问题



知识基础 AFLGo

- 定向模糊测试开山之作-AFLGo：两次编译、距离计算、种子排序
  - 编译->计算各基本块距离（静态分析）->插桩编译->fuzz（动态分析）
  - 调和平均数计算当前-多目标间的平均最短距离（Dijkstra算法）
  - 模拟退火算法引入随机性，避免种子排序过程中陷入局部最优
  - 对“高价值”种子分配更多测试资源（时间、变异优先级）





研究现状

- 软件灰盒定向模糊测试方法
  - 通过适当的调度方法，使模糊测试资源集中于预设的位置，并探索其附近的崩溃
  - 适用场景
    - 补丁验证、敏感功能测试
    - 漏洞复现与验证
    - 探索程序深层次漏洞
  - 面临问题
    - 种子排序方面：路径计算方法较简单，易将不可达的种子排在靠前位置
    - 插桩方面：大范围插桩，影响执行效率
    - 变异策略方面：沿用AFL的通用变异策略，缺乏针对目标路径的定向变异引导

SELECTFUZZ: Efficient Directed Fuzzing with Selective Path Exploration

SELECTFUZZ TIPO

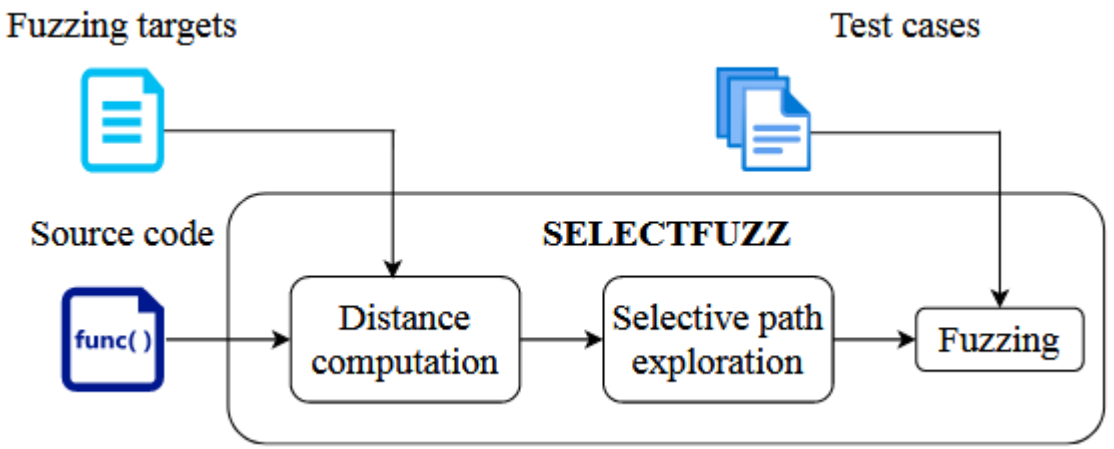
T	目标	针对性生成能到达待测程序的特定位置的输入，验证是否有潜在的漏洞
I	输入	待测程序源代码、测试目标位置
P	处理	1.编译源代码并向相关代码块插桩 2.收集测试反馈数据，计算现有输入距离目标代码块的距离 3.排序已有输入，针对可扩展覆盖率的输入优先变异 4.重复2-3操作，直到触发漏洞
O	输出	崩溃信息（是/否）、引起崩溃的输入

P	问题	现有定向模糊测试方法倾向于高覆盖率，忽略代码块与目标的关联性
C	条件	待测目标为非状态转移的程序，不会记录之前输入
D	难点	如何精准感知是否延高效的路径变异；如何使输入快速达到目标位置
L	水平	2023 IEEE Symposium on Security and Privacy (SP)

SELECTFUZZ 算法原理

- 核心创新：1个衡量标准，1个选取方式
  - 当前位置到目标的距离衡量标准(Distance computation)
    - 估算从一个基本块到目标代码，各个到达路径的“概率”，衡量各触发路径难度
    - 对高概率目标（相对较简单的路径），分配更多测试资源
  - 指导模糊测试器的选择性路径探索框架(Selective Path Exploration)
    - 相关代码块识别：路径分裂代码(path-divergent)和数据依赖代码(data-dependent)
    - 输入优先级排序：选取并优先测试更可能触发目标的输入
    - 目标调度：变异输入并进行测试
- 具体实现
  - 基于AFLGo改进
  - 选择性插桩，只对待测相关代码插桩

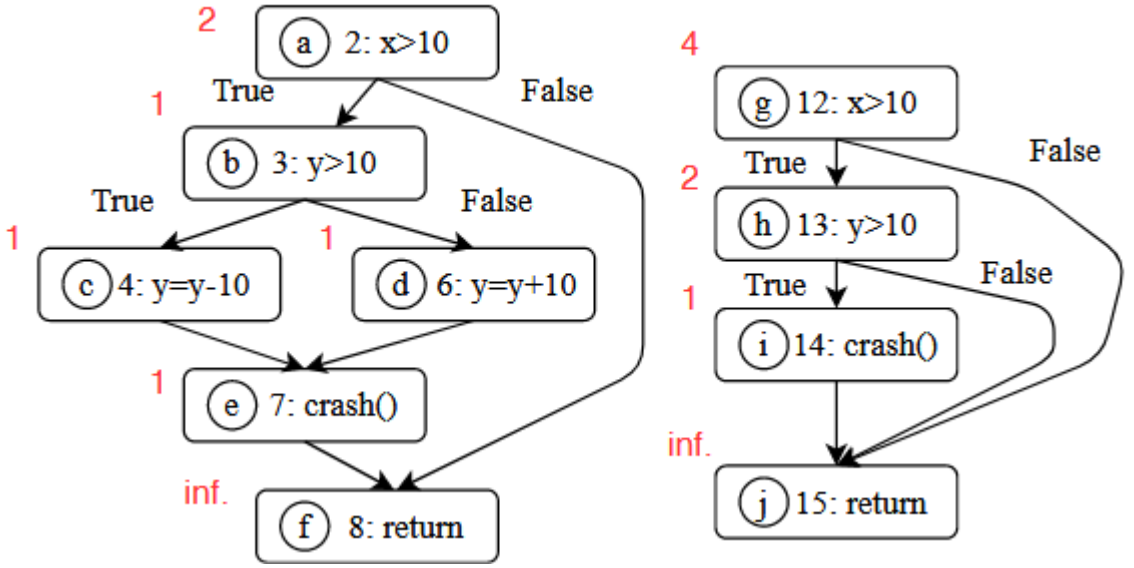
精准定向，高效分配



SELECTFUZZ 算法原理—距离衡量标准

- 基本块间距离
  - 根据控制流图(CFG)计算当前节点的所有分支平均值的倒数
  - $d(i,T) = \frac{n}{\sum_j^n d(i_j,T)}$，考虑所有可能的路径，全面评估概率
- 改进方面：AFLGo使用的Dijkstra算法衡量最短距离
  - 更精确获取当前代码块到待测代码的距离，以分配测试资源

```
1  void foo(x,y) {
2    if(x>10) {
3      if(y>10)
4        y=y-10;
5      else
6        y=y+10;
7      crash();
8    }
9    return;
10 }
11
12 void bar(x,y) {
13   if(x>10)
14     if(y>10)
15       crash();
16   return;
17 }
```



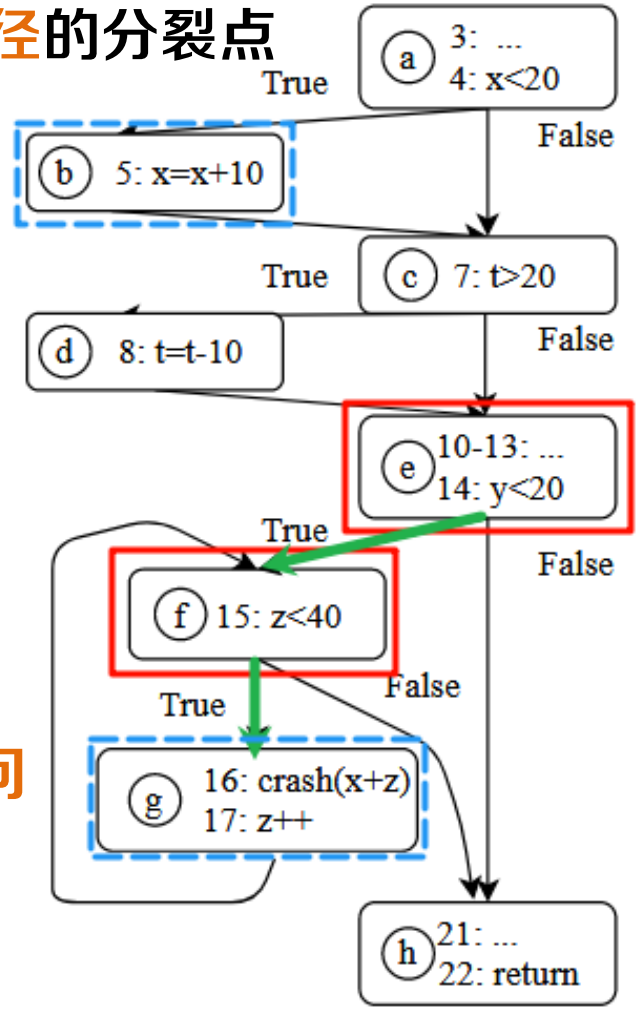
Algorithm 1 Block distance calculation.

Input : T, the target code locations;
Output : $dist$, a map to store the block distance.

$dist \leftarrow \{\}$
$prob \leftarrow \{\}$
$bb\_set \leftarrow build\_ICFG()$ // build the inter-procedural CFG and return all basic blocks
for b in bb_set **do**
  $b.status \leftarrow 0$
  $prob[b] \leftarrow 0$
end for
for b in bb_set **do**
  $dist[b] \leftarrow cal\_dist(b)$
end for
function cal_dist(BB b)
  $p = cal\_prob(b)$
  if $p == 0$ **then**
    return ∞
  else
    return $1/p$
  end if
end function
function cal_prob(BB b)
  if $b.status == 0$ **then**
    $b.status \leftarrow 1$
    if b in T **then**
      $prob[b] \leftarrow 1$
    else
      $sum \leftarrow 0$
      for $succ$ in $b.Successors$ **do**
        $prob[succ] \leftarrow cal\_prob(succ)$
        $sum \leftarrow sum + prob[succ]$
      end for
      $num \leftarrow b.getNumSuccessors()$
      if $num > 0$ **then**
        $prob[b] \leftarrow sum/num$
      end if
    end if
    $b.status \leftarrow 2$
  end if
  return $prob[b]$
end function

SELECTFUZZ 算法原理—选择性路径探索框架

- 相关代码块识别：定位和测试目标强相关的代码
  - 路径分裂代码块(path-divergent)：可达路径和不可达路径的分裂点
    - 仅对可达路径插桩，引导模糊测试延有效路径变异
  - 数据依赖代码块(data-dependent)：影响触发点变量值
    - 重点测试变量各种组合方式，触发潜在漏洞
- 输入优先级排序：优先变异、测试
  - 1.触发更多相关性代码块的输入
  - 2.连接已有相关代码块的输入
  - 3.距离测试目标较近的输入
- 目标调度：对高价值输入进行测试并指导后续变异方向
  - 沿用AFLGo的模拟退火算法



SLELCTFUZZ 实验设置

- 对比方法
  - AFLGo、Beacon（基于剪枝，未开源，作者自行复现）、Beacon+SELECTFUZZ
- 测试数据集
  - 18个已知CVE漏洞、Google Fuzzer Test Suite
- 评价指标
  - 漏洞发现平均时间(TTE)，p-value（越小说明统计结果越具有显著差异）
  - BBtotal：基本块总数、BBrec：可达基本块数量、 BBrel： 相关基本块数量
- 硬件环境
  - Intel Xeon(R) CPU W-2123 @ 3.60GHz、16G内存（远弱于实验室GPU服务器）
- 最大测试时间120小时

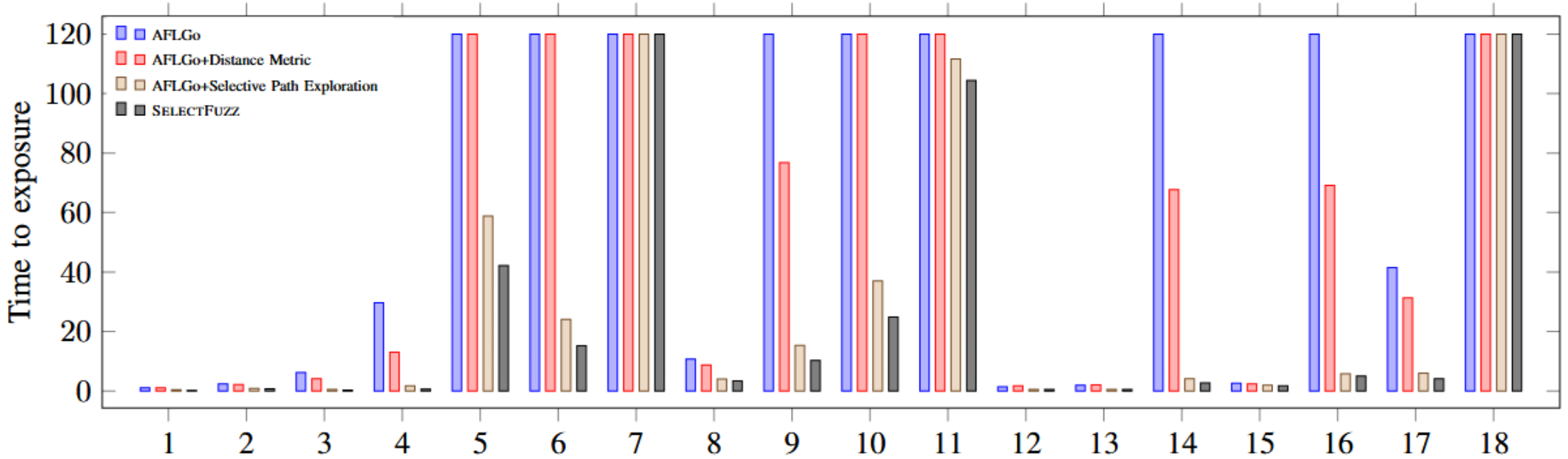
SLELCTFUZZ 对比实验

- RQ1: SELECTFUZZ触发已知漏洞的有效性
  - 通过路径探索策略提升定向模糊测试的效率，多测出7个崩溃
  - 在5个漏洞中测试中，提速超过10倍，最高达46.41倍
  - 可与路径剪枝策略互补，提升定向模糊测试效率

No.	CVE-ID	Program	Vuln. Code	BB_{total}	BB_{rec}	BB_{rel}	AFLGo	SELECTFUZZ	Beacon†	SELECTFUZZ*	Speedup	
											SELECTFUZZ	Beacon
1	2016-9827	swftophp-0.4.7	outputtxt.c:144	4,114	717	192	1.07 h	0.25 h	0.46 h	0.22 h	**4.28**	3.90
2	2017-7578	swftophp-0.4.7	parser.c:68	3,788	323	116	2.41 h	0.77 h	0.83 h	0.70 h	3.13	**8.10**
3	2018-8807	swftophp-0.4.8	decompile.c:349	3,798	626	298	6.23 h	0.32 h	3.74 h	0.27 h	**19.47**	5.67
4	2018-8962	swftophp-0.4.8	decompile.c:398	3,798	516	198	29.70 h	0.64 h	4.81 h	0.33 h	**46.41**	18.37
5	2017-11728	swftophp-0.4.8	decompile.c:868	3,798	484	221	120 h	42.17 h	32.10 h	19.87 h	2.85	**10.76**
6	2018-20427	swftophp-0.4.8	decompile.c:425	3,798	91	56	120 h	15.26 h	12.78 h	5.14 h	7.86	**35.10**
7	2017-8846	lrzip-0.631	stream.c:1756	9,454	511	325	120 h	120 h	120 h	120 h	1	N.A.
8	2016-4491	cxxfilt-2.26	cp-demangle.c:4318	42,150	414	163	10.73 h	3.38 h	4.13 h	3.09 h	3.17	**3.87**
9	2017-9047	xmllint-20904	valid.c:1279	69,696	18,032	1,094	120 h	10.28 h	42.30 h	9.92 h	**11.67**	6.66
10	2017-9048	xmllint-20904	valid.c:1323	69,696	18,046	1,021	120 h	24.87 h	45.57 h	16.47 h	4.83	**6.01**
11	2017-9049	xmllint-20902	parser.c:3406	69,853	20,128	4,540	120 h	104.39 h	71.83 h	33.04 h	1.15	**3.62**
12	2017-8392	objdump-2.28	dwarf2.c:4212	60,482	463	331	1.44 h	0.51 h	0.40 h	0.24 h	2.82	N.A.
13	2017-8396	objdump-2.28	libbfd.c:615	60,518	13,123	3,135	1.98 h	0.53 h	0.65 h	0.44 h	3.74	N.A.
14	2017-8397	objdump-2.28	reloc.c:885	60,518	2,800	697	120 h	2.73 h	43.20 h	2.03 h	43.96	N.A.
15	2017-14940	objdump-2.28	parser.c:3406	60,482	11,601	2,418	2.63 h	1.71 h	2.46 h	1.31 h	1.53	N.A.
16	2019-10872	libpoppler.so.87	Splash.cc:5872	84,662	2,028	91	120 h	5.02 h	59.30 h	4.84 h	23.90	N.A.
17	2019-10873	libpoppler.so.86	SplashXPathScanner.cc:458	84,529	3,942	281	41.52 h	4.13 h	21.25 h	4.03 h	9.95	N.A.
18	2019-14494	libpoppler.so.89	SplashOutputDev.cc:4584	84,925	2,231	111	120 h	120 h	120 h	118.89 h	1	N.A.

SELECTFUZZ 消融实验

- RQ2: SELECTFUZZ各个组件对性能的影响（测试目标与RQ1相同）
  - 距离度量使AFLGo效率平均提升39%，尤其复杂调用（如间接调用）效果显著
    - AFLGo距离计算不准确导致误判路径复杂度，将大量测试资源用于复杂分支
  - 选择性路径探索使AFLGo平均提升6.68倍，有效减少了无关代码的探索开销
  - SELECTFUZZ可以使模糊测试更精准的选择到达路径，从而提升效率



SLELCTFUZZ 性能分析实验

- RQ3: 影响SELECTFUZZ效率的因素
  - 插桩开销：仅对相关代码块插桩，几乎不影响运行速度（AFLGo降低57%）
  - 相关代码探索：快速优化种子文件(No.16，种子55kb)，将测试缩小到相关代码块
- RQ4: SELECTFUZZ测试BranchMark效果
  - 在8个目标中表现最优，可发现复杂程序libarchive中的漏洞

Program	Vuln. Code	AFLGo		Beacon		ParmeSan		Angora		AFLChurn		SELECTFUZZ	
		TTE	p	TTE	p	TTE	p	TTE	p	TTE	p	TTE	p
boringssl	asn1_lib.c:459	T.O.	-	T.O.	-	**19.92 h**	0.043	T.O.	-	T.O.	-	T.O.	-
c-ares	ares_create_query.c:196	<0.01 h	0.011	<0.01 h	0.002	<0.01 h	0.008	<0.01 h	0.075	<0.01 h	0.006	<0.01 h	0.002
guetzli	output_image.cc:398	0.58 h	0.006	0.50 h	0.003	0.17 h	0.006	0.62 h	0.028	0.47 h	0.007	**0.09 h**	0.004
harfbuzz	hb-buffer.cc:419	T.O.	-	22.11 h	0.008	**16.08 h**	0.004	T.O.	-	T.O.	-	20.40 h	0.015
json	fuzzer-parse_json.cpp:50	0.07 h	0.006	**0.03 h**	0.000	0.04 h	0.002	0.07 h	0.006	0.06 h	0.005	0.05 h	0.001
lcms	cmsintrp.c:642	8.38 h	0.046	**4.97 h**	0.001	8.18 h	0.006	22.90 h	0.107	7.42 h	0.006	6.23 h	0.004
lcms	cmsintrp.c:642†	6.90 h	0.005	5.77 h	0.006	5.58 h	0.018	8.86 h	0.079	7.32 h	0.006	**3.52 h**	0.004
libarchive	archive_read_support_format_warc.c:537	T.O.	-	T.O.	-	13.04 h	0.037	18.06 h	0.219	T.O.	-	**8.12 h**	0.034
libssh	messages.c:1001	0.21 h	0.004	0.52 h	0.132	0.22 h	0.008	0.20 h	0.005	0.12 h	0.004	**0.09 h**	0.002
libxml2	parser.c:10666	0.09 h	0.006	0.02 h	0.002	**<0.01 h**	0.000	0.42 h	0.002	**<0.01 h**	0.001	**<0.01 h**	0.000
libxml2	dict.c:489	0.33 h	0.005	0.26 h	0.004	**0.08 h**	0.004	0.49 h	0.007	0.19 h	0.001	0.10 h	0.002
openssl-1.0.1f	t1_lib.c:2586	<0.01 h	0.001	<0.01 h	0.000	<0.01 h	0.000	<0.01 h	0.001	<0.01 h	0.002	<0.01 h	0.000
openssl-1.0.2d	target.cc:145	0.07 h	0.002	**0.02 h**	0.000	0.05 h	0.006	0.12 h	0.018	0.08 h	0.006	0.05 h	0.000
pcre	pcre2_match.c:1426	0.62 h	0.005	0.56 h	0.002	0.60 h	0.003	1.34 h	0.008	0.53 h	0.006	**0.40 h**	0.006
re2	nfa.cc:532	22.43 h	0.036	10.90 h	0.009	13.12 h	0.004	T.O.	-	13.46 h	0.018	**9.57 h**	0.011
vorbis	codebook.c:407	T.O.	-	**21.55 h**	0.040	T.O.	-	T.O.	-	T.O.	-	T.O.	-
woff	woff2_dec.cc:1274	6.47 h	0.006	6.04 h	0.003	4.93 h	0.001	12.57 h	0.028	6.32 h	0.006	**4.56 h**	0.004

SELECTFUZZ 实战效果

- RQ5: SELECTFUZZ发现真实bug能力测试
  - 针对待测程序中已披露的CVE漏洞附近代码测试
  - 共检测出14个未知漏洞，包括堆溢出、段错误、内存泄漏等
  - 已有11个漏洞被修复，其中6个漏洞分配了新CVE编号，1个已经隐藏3年

Project	Root Cause	Vul. Type	Status	CVE
poppler	Object.h:435	abort	patched	issue-1274
poppler	XRef.cc:1388	abort	patched	issue-1278
poppler	Object.h:435	abort	patched	issue-1276
poppler	PDFDoc.cc:1755	abort	patched	issue-1282
poppler	Object.h:445	abort	patched	issue-1289
poppler	pdftocairo.cc:731	assertion	confirmed	issue-1287
libjpeg	iostream.cpp:543	infinite loop	patched	2022-37768
libjpeg	losslessscan.cpp:374	seg. fault	patched	2022-37769
libjpeg	linemerger.cpp:262	seg. fault	patched	2022-37770
tcpreplay	get.c:344	heap overflow	patched	2022-37048
tcpreplay	get.c:150	heap overflow	patched	2022-37049
tcpreplay	get.c:713	heap overflow	patched	2022-37047
libtiff	tif_jpeg.c:962	assertion	confirmed	issue-445
libming	parser.c:2431	memory leak	-	issue-239

DeepGo: Predictive Directed Greybox Fuzzing

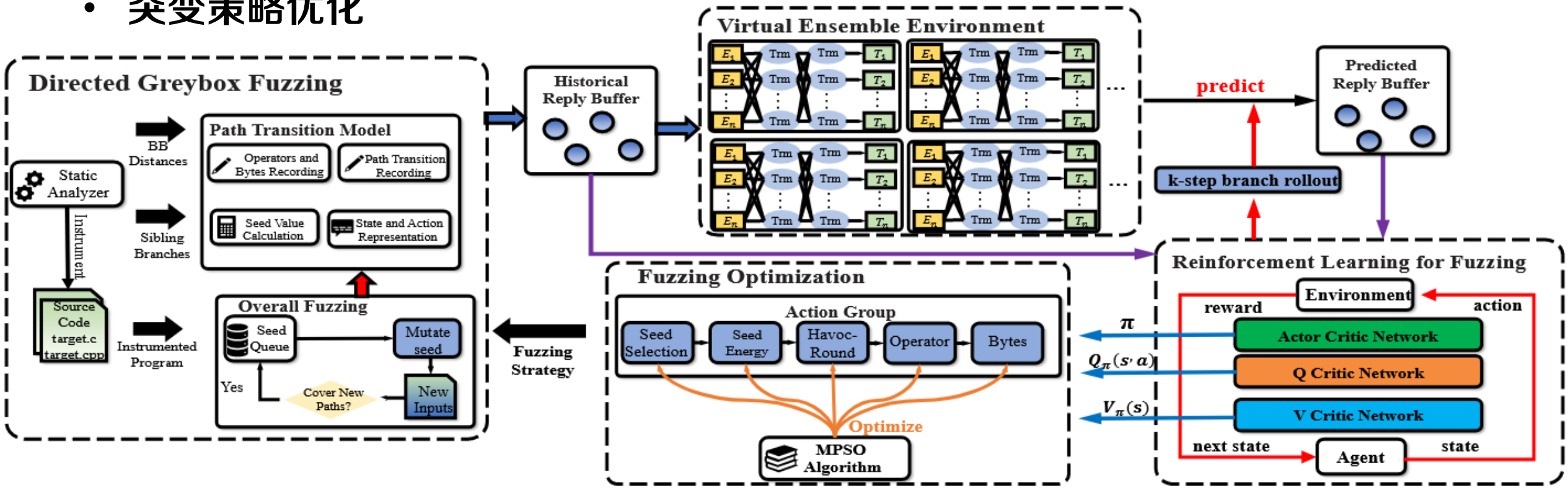
DeepGo TIPO

T	目标	针对性生成能到达待测程序的特定位置的输入，验证是否有潜在的漏洞
I	输入	待测程序源代码、测试目标位置
P	处理	1. 构建路径覆盖数据集并训练预测模型 2. 基于强化学习和预测模拟，生成变异策略 3. 应用策略引导灰盒模糊测试并持续反馈训练
O	输出	崩溃信息（是/否）、引起崩溃的输入

P	问题	现有定向模糊测试依赖已有的输入，对未触发的路径无法预测
C	条件	待测目标为非状态转移的程序，不会记录之前输入
D	难点	如何利用现有的输入、路径对应关系预测未知路径的触发方式
L	水平	2024 Network and Distributed System Security(NDSS) Symposium

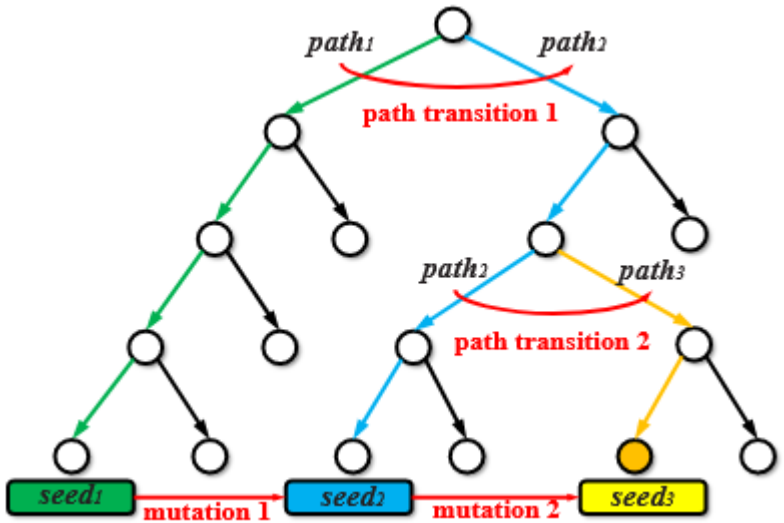
DeepGo 算法原理

- 路径转换模型：将fuzz建模为通过特定的路径转换序列到达目标点
- 奖励预测模型：在不执行种子的情况下预测潜在的路径转换和相应的奖励
- 强化学习模型：结合历史及预测的路径转换，学习最大化奖励的路径转换策略
- 突变策略优化



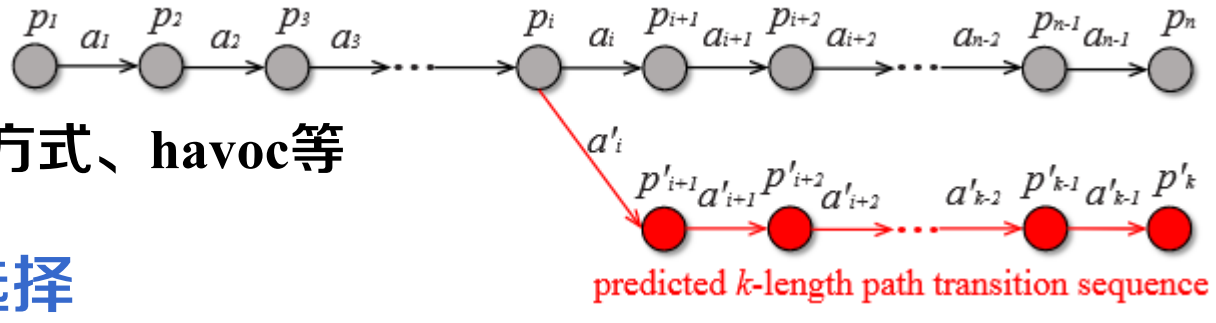
DeepGo 算法原理—路径转换模型

- 路径转换难度计算
  - $P(ubr) = \frac{1}{\sum_{br \in \phi(cond)} hit_{br}+1}$，类似蒙特卡洛方法估计转换难度（改进算法1）
- 种子价值Vs=当前路径到目标的距离+ $P(ubr)$ +执行速度+种子是否被标记
- 变异造成的种子价值差异$r(p_t, a_t, p_{t+1}) = V_s(p_t + 1) - V_s(p_t)$
- 路径转换对到达目标点的贡献
  - $Q_\pi(p, a) = E_{p' \sim P}[r(p, a, p') + \gamma V_\pi^t(p')]$
- 策略π下，路径p选择动作a的概率
  - $\sum_a \pi(a|p) \cdot Q_\pi(p', a)$



DeepGo 算法原理

- 奖励预测模型：根据输入预测结果
  - 使用DNN预测不同输入造成的路径转换行为
  - 损失函数$\mathcal{L}(\theta)$，衡量DNN输出与真实标签的关系，训练模型以最小化损失
    - $\mathcal{L}(\theta)=\sum_{n=1}^{N}[\mu_\theta(s_n,a_n)-s_{n+1}]^T\Sigma_\theta^{-1}(s_n,a_n)[\mu_\theta(s_n,a_n)-s_{n+1}]+logdet\Sigma_\theta(s_n,a_n)$
  - 预测对某个路径p_t进行某种变异a_t后，会转移到哪条新路径p_{t+1}，并估计价值
- 动作选择模型：基于强化学习选择能最大化奖励的突变动作
  - 根据原始路径$a_1,\dots,a_i,\dots,a_{n-1}$，目标路径$a_1,\dots,a_i',\dots,a_{k-1}'$
  - 尝试不同的a_t，使用奖励预测模型获取执行路径
  - 提供变异策略
    - 如种子排序、变异位置选择、变异方式、havoc等



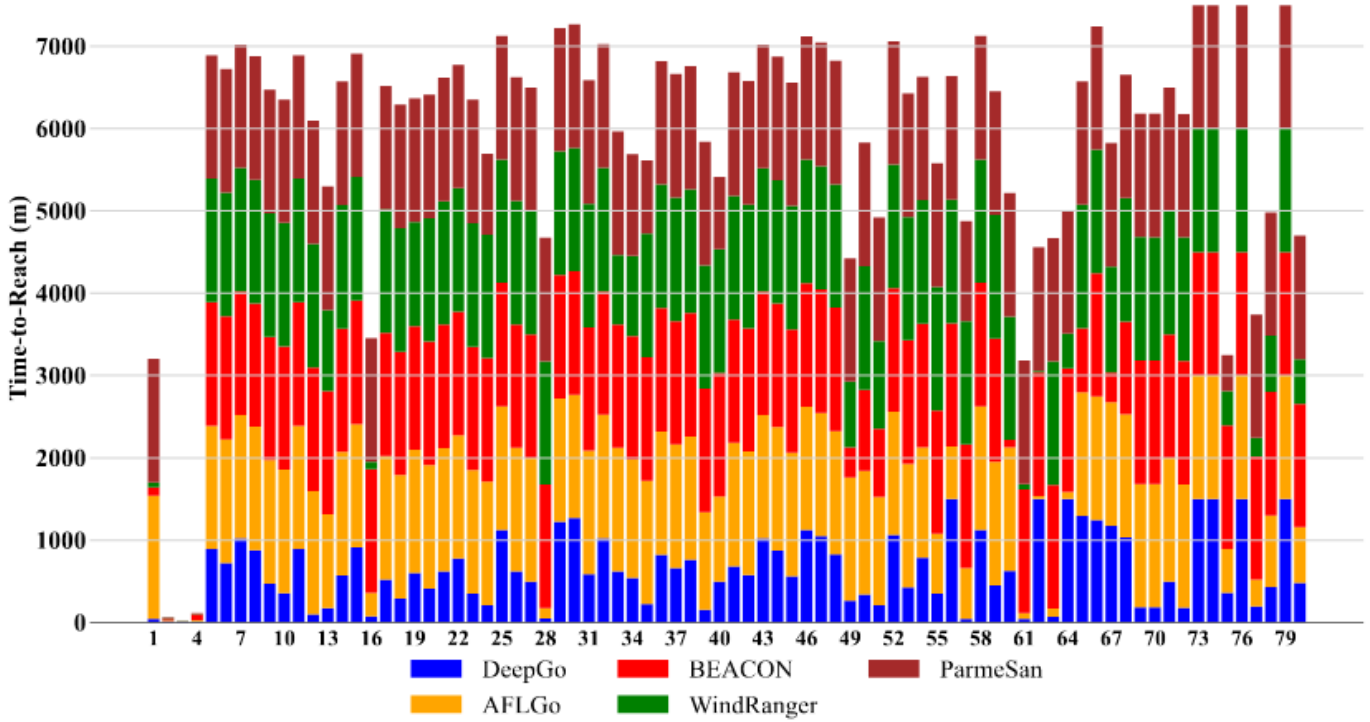
输入期望的路径，提供变异策略选择

DeepGo 实验设计

- 对比方法
  - AFLGo、Beacon、WindRanger、ParmeSan
- 测试数据集
  - 20个已知CVE漏洞、UniBench
- 评价指标
  - 漏洞发现平均时间(TTE)，p-value
  - Vargha-Delaney测度$\hat{A}_{12}$（越趋近1说明前者效果越好），大于0.71时称为显著好于
- 消融实验
  - DeepGo-r：删除DeepGo的强化学习模块和优化组件模块
  - DeepGo-v：删除DeepGo的奖励预测模型

DeepGo 对比实验

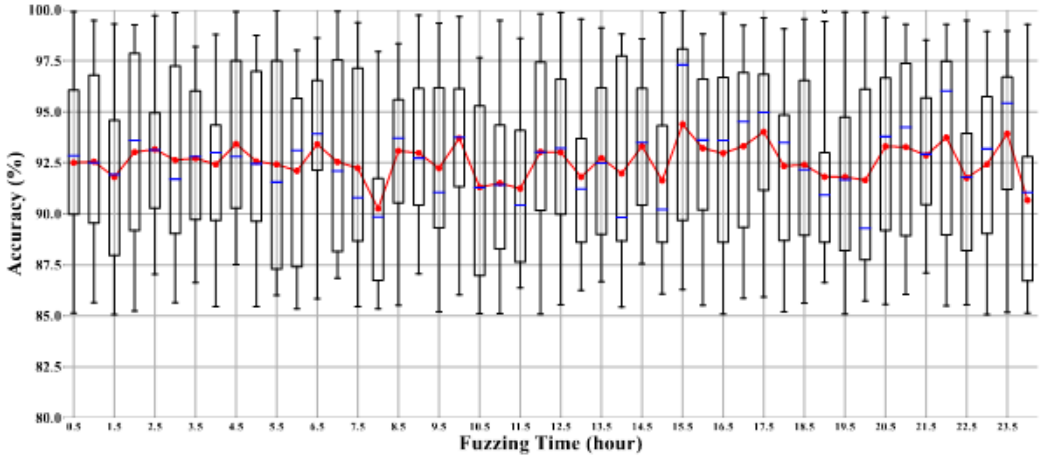
- RQ1: DeepGo到达特定测试代码方面性能
  - 首次触发目标代码的TTR最低提升1.72倍
- RQ2: DeepGo触发已知漏洞的有效性
  - 挖掘漏洞的TTR最低提升2.43倍，最低$\hat{A}_{12}$为0.72



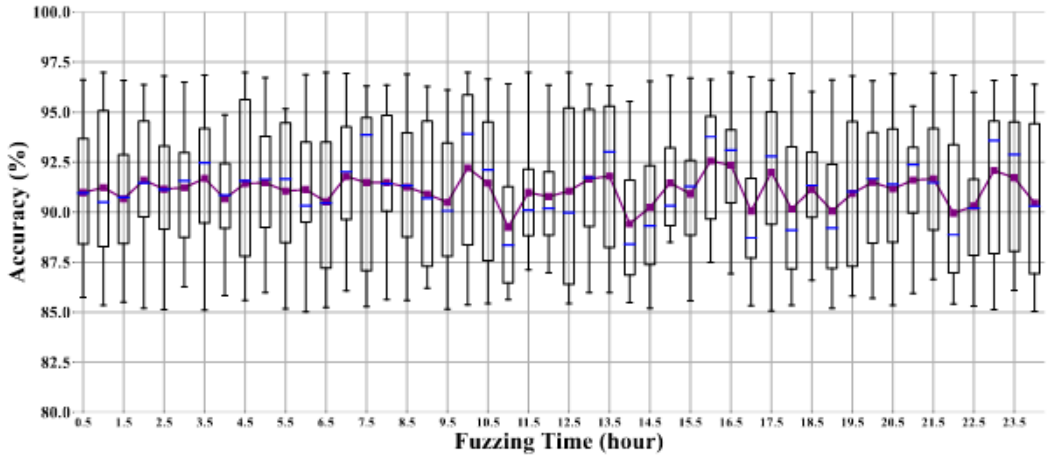
Prog.	CVE-ID	AFLGo	BEACON	WindRa	ParmeS	DeepGo
$\text{binutils}_{2.26}$	2016-4487	2.33m	0.63m	1.21m	0.95m	1.34m
	2016-4488	4.23m	32.1m	3.32m	2.62m	2.69m
	2016-4489	3.36m	2.98m	5.88m	2.31m	1.23m
	2016-4490	1.15m	2.35m	2.63m	0.82m	1.97m
	2016-4491	448m	258m	298m	212m	129m
	2016-4492	10.8m	43.6m	7.47m	4.33m	6.94m
	2016-6131	348m	292m	318m	244m	68.1m
$\text{libming}_{4.48}$	2018-8807	331m	267m	171m	301m	101m
	2018-8962	234m	163m	121m	198m	54.8m
	2018-11095	T.O.	914m	1311m	T.O.	812m
	2018-11225	T.O.	438m	996m	T.O.	128m
$\text{LibPNG}_{1.5.1}$	2011-2501	10.2m	N/A	7.81m	4.53m	3.46m
	2011-3328	69.1m	N/A	49.3m	193m	17.5m
	2015-8540	0.88m	N/A	0.96m	3.41m	5.65m
$\text{xmllint}_{2.9.4}$	2017-9047	T.O.	T.O.	T.O.	T.O.	783m
	2017-9048	T.O.	T.O.	T.O.	T.O.	1389m
	2017-9049	T.O.	T.O.	T.O.	T.O.	T.O.
	2017-9050	T.O.	T.O.	T.O.	T.O.	911m
$\text{Lrzip}_{0.631}$	2017-8846	348m	156m	223m	466m	131m
	2018-11496	201m	98.1m	169m	126m	78.9m
speedup		**2.61×**	**3.32×**	**2.43×**	**2.53×**	-
mean $\hat{A}_{12}$		**0.79**	**0.72**	**0.75**	**0.81**	-
mean p-values		0.018	0.032	0.026	0.011	-

DeepGo 对比实验

- RQ3: 预测概率和路径转换的准确率如何?
  - 路径转换的预测概率(AAPP)平均精度，预测奖励(AAPR)的平均精度
  - 从0.5小时到24小时，AAPP和AAPR的准确率均大于80%
  - AAPP和AAPR在48个时间点的平均值分别为92.57%和91.10%
  - 这表明奖励预测模型可以很好预测路径转换概率和奖励



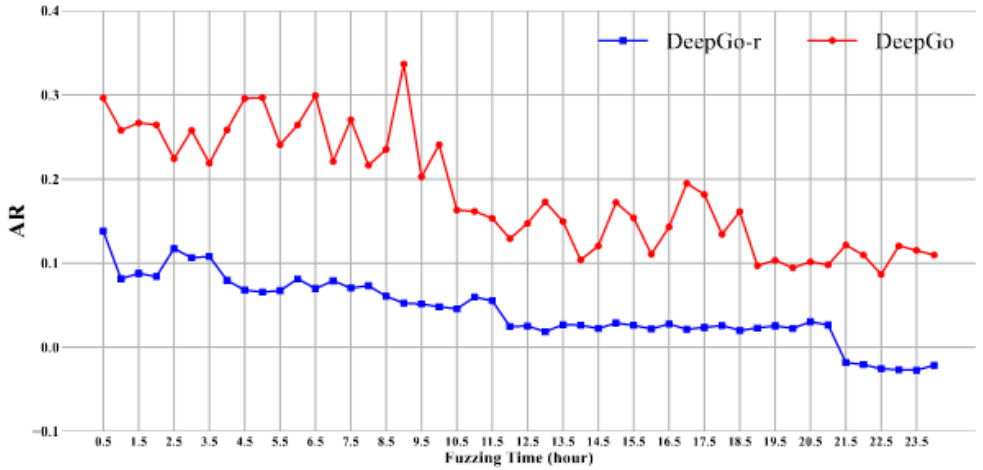
(a) AAPP



(b) AAPR

DeepGo 消融实验

- RQ4：强化学习模块是否有效
  - DeepGo的平均奖励(AR)平均比DeepGo-r的AR高4.26倍
  - 强化学习模块可以引导模糊器得出最优的路径转换序列，更快地到达目标位置
- RQ5：DeepGo各模块对发现漏洞效率方面的影响
  - DeepGo(73/80)分别比DeepGo-v(32/80)和DeepGo-r(18/80)可以到达更多的测试目标
  - DeepGo的平均TTR上分别比DeepGo-v和DeepGo-r高出2.05倍和3.72倍



No	Prog	Version	Target sites	AFLGo	BEACON	WindRanger	ParmeSan	DeepGo-v	DeepGo-r	DeepGo
1	cflow	1.6	parser.c:281	T.O.	99.4m	61.1m	T.O.	58.1m	T.O.	41.8m
2			c.c:1783	12.8m	22.1m	6.45m	10.1m	12.2m	16.4m	11.2m
3			parser.c:1223	1.22m	0.83m	2.44m	6.23m	6.16m	4.76m	8.23m
4			parser.c:108	12.8m	68.1m	8.32m	6.76m	13.5m	17.2m	16.1m
5	mp42aac	Bento4 1.5.1-628	Ap4AvccAtom.cpp:82	T.O.	N/A	T.O.	N/A	T.O.	T.O.	891m
6			Ap4TrunAtom.cpp:139	T.O.	N/A	T.O.	N/A	T.O.	T.O.	723m
7			Ap4SbgpAtom.cpp:81	T.O.	N/A	T.O.	N/A	T.O.	T.O.	1022m
8			Ap4AtomFactory.cpp:490	T.O.	N/A	T.O.	N/A	T.O.	T.O.	878m
9	jhead	3.00	exif.c:1339	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	472m
10			iptc.c:143	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	355m
11			iptc.c:91	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	892m
12			makernote.c:174	T.O.	N/A	T.O.	T.O.	671m	T.O.	98.1m
13	mp3gain	1.5.2	layer3.c:1116	1142m	N/A	984m	N/A	871m	T.O.	172m
14			mp3gain.c:602	T.O.	N/A	T.O.	N/A	T.O.	T.O.	572m
15			interface.c:663	T.O.	N/A	T.O.	N/A	T.O.	T.O.	912m
16			apetag.c:341	290m	N/A	91.2m	N/A	164m	345m	72.8m

特点总结与未来展望

- 特点总结

算法	SELECTFUZZ	DeepGo
优势	1.更全面的距离计算方法 2.定位关联代码块，选择性的插桩方式 3.算法简洁，发现漏洞效率高	1.结合历史信息和预测信息，引导定向模糊测试选择最优路径到达目标站点 2.通过预测实现变异策略选择 3.注重“道”的创新，领域首次使用强化学习
劣势	1.路径概率计算忽略分支条件，可能导致分支难度误判 2.会探索所有可达路径，但其中并非所有路径都存在并可以触发漏洞，效率较低	1.动作空间建模较简单，仅关联状态转移和动作类型，忽略具体修改位置和内容 2.DNN对于离散空间的预测效果不佳 3.强化学习需要收敛时间，策略更新延迟

- 未来展望
  - 智能化：定向模糊测试目标具有明显的功能特征，利用大模型分析提供精准约束
  - 变异方式：引入符号执行加速复杂分支的求解，提升覆盖率和触发可能性
  - 可解释性：发现Bug后自动定位、分析、修复，提升开发测试人员效率

参考文献

[1] Luo C, Meng W, Li P. Selectfuzz: Efficient directed fuzzing with selective path exploration[C]//2023 IEEE Symposium on Security and Privacy (SP). IEEE, 2023: 2693-2707.

[2] https://github.com/cuhk-seclab/SelectFuzz

[3] Lin P, Wang P, Zhou X, et al. DeepGo: Predictive Directed Greybox Fuzzing[C]//Proceedings of the Network and Distributed System Security Symposium. 2024.

[4]徐邑江,高庆,陈立果,等.定向灰盒模糊测试技术研究进展[J].计算机学报,2025,48(05):1244-1272.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！

算法2—RQ5完整实验结果

- 算法2-RQ5完整实验结果

No	Prog	Version	Target sites	AFLGo	BEACON	WindRanger	ParmeSan	DeepGo-v	DeepGo-r	DeepGo
1	cflow	1.6	parser.c:281	T.O.	99.4m	61.1m	T.O.	58.1m	T.O.	41.8m
2			c.c:1783	12.8m	22.1m	6.45m	10.1m	12.2m	16.4m	11.2m
3			parser.c:1223	1.22m	0.83m	2.44m	6.23m	6.16m	4.76m	8.23m
4			parser.c:108	12.8m	68.1m	8.32m	6.76m	13.5m	17.2m	16.1m
5	mp42aac	Bento4 1.5.1-628	Ap4AvccAtom.cpp:82	T.O.	N/A	T.O.	N/A	T.O.	T.O.	891m
6			Ap4TrunAtom.cpp:139	T.O.	N/A	T.O.	N/A	T.O.	T.O.	723m
7			Ap4SbgpAtom.cpp:81	T.O.	N/A	T.O.	N/A	T.O.	T.O.	1022m
8			Ap4AtomFactory.cpp:490	T.O.	N/A	T.O.	N/A	T.O.	T.O.	878m
9	jhead	3.00	exif.c:1339	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	472m
10			iptc.c:143	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	355m
11			iptc.c:91	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	892m
12			makernote.c:174	T.O.	N/A	T.O.	T.O.	671m	T.O.	98.1m
13	mp3gain	1.5.2	layer3.c:1116	1142m	N/A	984m	N/A	871m	T.O.	172m
14			mp3gain.c:602	T.O.	N/A	T.O.	N/A	T.O.	T.O.	572m
15			interface.c:663	T.O.	N/A	T.O.	N/A	T.O.	T.O.	912m
16			apetag.c:341	290m	N/A	91.2m	N/A	164m	345m	72.8m
17	lame	3.99.5	bitstream.c:823	T.O.	N/A	T.O.	N/A	T.O.	T.O.	521m
18			lame.c:2148	T.O.	N/A	T.O.	N/A	T.O.	T.O.	291m
19			uantize_pvt.c:441	T.O.	N/A	1269m	N/A	T.O.	T.O.	599m
20			get_audio.c:1605	T.O.	N/A	T.O.	N/A	932m	T.O.	412m
21	imginfo	jasper 2.0.12	jp2_cod.c:841	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	619m
22			jp2_cod.c:636	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	776m
23			jas_stream.c:823	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	351m
24			jpc_dec.c:1393	T.O.	N/A	T.O.	984m	641m	T.O.	211m
25	gdk-pixbuf-pixdata	gdk-pixbuf 2.31.1	gdk-pixbuf-loader.c:387	T.O.	T.O.	T.O.	N/A	T.O.	T.O.	1126m
26			io-qtif.c:511	T.O.	T.O.	T.O.	N/A	T.O.	T.O.	622m
27			io-jpeg.c:691	T.O.	T.O.	T.O.	N/A	T.O.	T.O.	498m
28			io-tga.c:360	126m	T.O.	T.O.	N/A	87.3m	172m	48.7m
29	jq	1.5	jv_dtoa.c:3122	T.O.	T.O.	N/A	T.O.	T.O.	T.O.	1223m
30			jv_dtoa.c:2004	T.O.	T.O.	N/A	T.O.	T.O.	T.O.	1267m
31			jv_dtoa.c:2518	T.O.	T.O.	N/A	T.O.	T.O.	T.O.	587m
32			jv_unicode.c:42	T.O.	T.O.	N/A	T.O.	T.O.	T.O.	1024m
33	tcpdump	4.8.1	print-aodv.c:259	T.O.	N/A	843m	T.O.	T.O.	T.O.	622m
34			print-ntp.c:412	1436m	N/A	974m	1239m	1213m	T.O.	542m
35			print-rsvp.c:1252	T.O.	N/A	T.O.	889m	338m	T.O.	223m
36			print-l2tp.c:606	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	821m
37	tic	ncurses 6.1	captoinfo.c:189	T.O.	N/A	N/A	T.O.	T.O.	T.O.	662m
38			alloc_entry.c:141	T.O.	N/A	N/A	T.O.	T.O.	T.O.	761m
39			name_match.c:111	1186m	N/A	N/A	T.O.	251m	1368m	155m
40			entries.c:78	1038m	N/A	N/A	883m	749m	141m	495m
41	flvmeta	1.2.1	json.c:1036	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	682m
42			api.c:718	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	577m
43			flvmeta.c:1023	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	1021m
44			check.c:769	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	874m
45	tiffsplit	libtiff 3.9.7	tif_ojpeg.c:1277	T.O.	N/A	T.O.	N/A	T.O.	T.O.	561m
46			tif_read.c:335	T.O.	N/A	T.O.	N/A	T.O.	T.O.	1123m
47			tif_jbig.c:277	T.O.	N/A	T.O.	N/A	T.O.	T.O.	1046m
48			tif_dirread.c:1977	T.O.	N/A	T.O.	N/A	1068m	T.O.	825m
49	nm	binutils-5279478	tekhex.c:325	T.O.	364m	798m	N/A	369m	T.O.	264m
50			elf.c:8793	T.O.	986m	T.O.	N/A	665m	T.O.	342m
51			dwarf2.c:2378	1313m	831m	1065m	N/A	942m	T.O.	212m
52			elf-properties.c:51	T.O.	T.O.	T.O.	N/A	T.O.	T.O.	1062m
53	pdftotext	4.00	XRef.cc:645	T.O.	N/A	T.O.	N/A	T.O.	T.O.	427m
54			GfxFont.cc:1337	1345m	N/A	T.O.	N/A	1143m	1423m	785m
55			Stream.cc:1004	725m	N/A	T.O.	N/A	498m	911m	352m
56			GfxFont.cc:1643	637m	N/A	T.O.	N/A	T.O.	876m	T.O.
57	sqlite3	SQLite 3.8.9	pager.c:5017	617m	N/A	N/A	1214m	325m	810m	44.1m
58			func.c:1029	T.O.	T.O.	N/A	T.O.	T.O.	T.O.	1126m
59			insert.c:1498	T.O.	T.O.	N/A	T.O.	310m	T.O.	452m
60			vdbe.c:1984	T.O.	89.6m	N/A	T.O.	T.O.	T.O.	628m
61	exiv2	0.26	tiffcomposite.cpp:82	73.1m	N/A	68.1m	N/A	57.2m	93.2m	42.1m
62			XMPMeta-Parse.cpp:847	37.5m	N/A	21.4m	N/A	T.O.	79.5m	T.O.
63			tiffvisitor.cpp:1044	102m	N/A	T.O.	N/A	89.2m	112m	69.5m
64			XMPMeta-Parse.cpp:896	86.7m	N/A	421m	N/A	T.O.	99.1m	T.O.
65	objdump	binutils-2.28	elf.c:9509	T.O.	782m	T.O.	T.O.	T.O.	T.O.	1294m
66			section.c:936	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	1244m
67			bfd.c:1108	T.O.	361m	1288m	T.O.	T.O.	T.O.	1175m
68			bfdio.c:262	T.O.	1123m	T.O.	T.O.	T.O.	T.O.	1032m
69	ffmpeg	4.0.1	rawdec.c:268	T.O.	N/A	N/A	T.O.	221m	T.O.	183m
70			decode.c:557	T.O.	N/A	N/A	N/A	338m	T.O.	182m
71			dump.c:632	T.O.	N/A	N/A	N/A	673m	T.O.	498m
72			utils.c	T.O.	N/A	N/A	N/A	238m	T.O.	178m
73	mujs	1.0.2	jsrun.c:572	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	T.O.
74			jsgc.c:47	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	T.O.
75			jsdump.c:292	532m	N/A	421m	433m	488m	T.O.	361m
76			jsvalue.c:362	T.O.	N/A	T.O.	T.O.	T.O.	T.O.	T.O.
77	swftools	0.9.2	initcode.c:242	324m	N/A	223m	N/A	298m	401m	196m
78			png.c:410	871m	N/A	681m	N/A	501m	1004m	431m
79			poly.c:137	T.O.	N/A	T.O.	N/A	T.O.	T.O.	T.O.
80			jpeg2swf.c:257	677m	N/A	541m	N/A	611m	881m	481m
		speedup		3.23×	1.72×	1.81×	4.83×	2.05×	4.26×	-
		mean $\hat{A}_{12}$		0.86	0.81	0.83	0.89	0.83	0.90	-
		mean p-values		0.008	0.032	0.016	0.001	0.013	0.006	-