基于指针流重排的并发漏洞检测技术

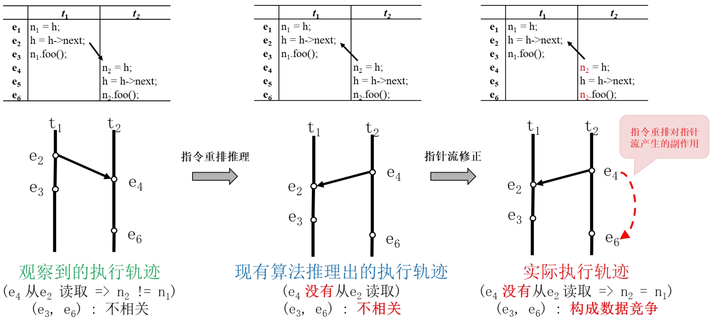
Reorder Pointer Flow in Sound Concurrency Bug Prediction

郭宇祺、祝世豪、蔡彦（**yancai@ios.ac.cn**）、和亮、张健
中国科学院软件研究所，计算机科学国家重点实验室

并发漏洞严重威胁着并发程序，然而，由于线程间的交错顺序随机，并发缺陷检测面临**状态空间爆炸**的问题。基于动态方法的精确并发漏洞检测技术通过在已有执行序列基础上推测可能的执行序列来检测并发漏洞，近年来，该类技术取得了长足的发展，先后涌现的多种方法取得了越来越好的效果。

然而，现有动态并发漏洞检测技术存在严重的局限性：未考虑**线程交错顺序对程序特定操作行为产生的影响**。这一局限性制约了现有动态检测技术对各类并发漏洞的检测能力。针对这一问题，本方法结合**流敏感**的**指针分析技术**，分析程序在**不同线程交错顺序**下可能具有的**不同指针流**，在保证**无误报**的情况下，通过考虑**线程交错顺序对程序指针流及其各类内存读写行为的影响**，提出了一种新型并发漏洞检测技术，实现了比现有最先进同类方法更好的并发漏洞检测效果。

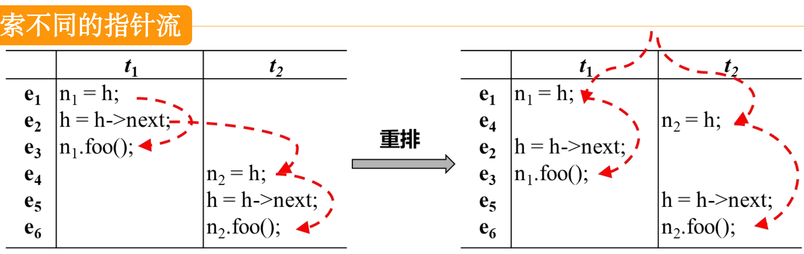
现有方法缺陷



给定图1所示的程序执行轨迹，现有并发漏洞检测技术通过对给定的程序执行轨迹进行指令重排推理，尝试找到一个能够触发并发漏洞的线程交错顺序（在这个例子中，为e_3和e_6事件之间的数据竞争漏洞）。然而，由于现有算法没有考虑到指令重排会对指针h以及n_2的值产生影响，并进一步影响事件e_6所操作的对象，因此现有技术会认为在重排后的程序执行轨迹中，n_1和n_2这两个指针将会始终保持如图1所示的不相关关系（即指向不同对象）。即使现有技术找到了如图2所示的程序执行轨迹，也无法捕获相应的并发漏洞。

为了解决以上问题，本方法引入流敏感的指针分析技术，在指令重排的过程中追踪重排对程序指针流产生的影响，并随时根据指针流的变化，修正程序的内存读写操作行为，从而实现了更准确的并发漏洞检测。在此基础上，本方法进一步提出主动探索能够引发指针流变动的指令重排策略，从而实现对线程交错空间更深程度的探索，达到了比现有技术更好的并发漏洞检测效果。

主动探索不同的指针流



实验验证

Program	Version	LOC	Thread	EAGLE	UFO	CONVUL	PERIOD	$EAGLE_n$
pbzip2	v0.9.4	45.9K	5	**5 NPD, 5 UPU, 8 UAF, 1 DF**	4 UAF	1 NPD, 1 UAF, 0 DF	0	**5 NPD, 5 UPU**, 4 UAF, 0 DF
pixz	v1.0.7	89.8K	6	0 NPD, **1 UPU**, 0 UAF, **1 DF**	0 UAF	0 NPD, 0 UAF, **1 DF**	0	0 NPD, **1 UPU**, 0 UAF, 0 DF
pigz	v2.7	107.1K	6	**8 NPD, 3 UPU, 4 UAF**, 0 DF	0 UAF	7 NPD, 0 UAF, 0 DF	0	**8 NPD, 3 UPU**, 3 UAF, 0 DF
lbzip2	v2.5	300.5K	7	0 NPD, **9 UPU**, 0 UAF, 0 DF	0 UAF	0 NPD, 0 UAF, 0 DF	0	0 NPD, 2 UPU, 0 UAF, 0 DF
httrack	v3.43.9	528.4K	2	0 NPD, **3 UPU**, 0 UAF, **1 DF**	0 UAF	#Err	0	0 NPD, **3 UPU**, 0 UAF, **1 DF**
libwebp	v1.2.2	822.2K	3	**4 NPD**, 0 UPU, 0 UAF, **2 DF**	0 UAF	1 NPD, 0 UAF, 0 DF	0	0 NPD, 0 UPU, 0 UAF, 0 DF
libvpx	v1.11.0	2.1M	4	**4 NPD, 31 UPU**, 0 UAF, 0 DF	#Err	**4 NPD**, 0 UAF, 0 DF	0	1 NPD, **31 UPU**, 0 UAF, 0 DF
x264	HEAD: 19856c	632.1K	7	**34 NPD, 59 UPU**, 0 UAF, **1 DF**	0 UAF	10 NPD, 0 UAF, 0 DF	0	2 NPD, 47 UPU, 0 UAF, **1 DF**
x265	v3.4	3.0M	10	0 NPD, **23 UPU**, 0 UAF, **1 DF**	0 UAF	0 NPD, 0 UAF, 0 DF	0	0 NPD, 0 UPU, 0 UAF, **1 DF**
MySQL	v5.7.36	10.9M	27	**17 NPD, 68 UPU**, 0 UAF, **16 DF**	0 UAF	0 NPD, 0 UAF, 0 DF	0	16 NPD, 63 UPU, 0 UAF, **16 DF**
Total	N/A	N/A	N/A	**72 NPD, 202 UPU, 12 UAF, 23 DF**	4 UAF	23 NPD, 1 UAF. 1 DF	0	32 NPD, 155 UPU, 7 UAF, 19 DF

Program	N_{rw}	N_l	N_m	N_f	N_{br}	EAGLE	UFO	SpeedUp	CONVUL	SpeedUp	PERIOD	CONPTA	Prop
pbzip2	1.57K	84	29	212	258	0.04s	4m52s	7.30×10^5	3.06s	7.65×10^1	>300h	0.01s	25.0%
pixz	31.48K	1.14K	2.17K	3.16K	4.72K	0.39s	33.19s	8.51×10^1	57.20s	1.47×10^2	>300h	0.02s	5.13%
pigz	25.27K	2.70K	1.36K	10.44K	33.41K	5.00s	20h59m21s	1.1×10^4	27.83s	5.57	>300h	0.01s	0.20%
lbzip2	126.65M	462	603	437.64K	16.73M	12m55s	3m20s	2.58×10^{-1}	12.20s	1.57×10^{-2}	>300h	3m13s	24.90%
httrack	19.76M	1.60K	66.10K	3.91M	26.82M	58.43s	0.39s	6.67×10^{-3}	#Err	#Err	>300h	8.57s	14.67%
libwebp	115.74M	564	1.69K	3.16M	5.07M	6m52s	141h41m59s	1.24×10^3	2m00s	2.91×10^{-1}	>300h	1m21s	19.63%
libvpx	98.56M	8.37K	11.12K	5.40M	38.88M	1h06m40s	#Err	#Err	25m26s	3.83×10^{-1}	>300h	1m16s	1.90%
x264	152.13M	7.13K	49.23K	4.86M	38.92M	1h03m30s	>300h	$>2.83\times10^2$	8m01s	1.26×10^{-1}	>300h	7m34s	11.93%
x265	140.33M	4.47K	11.52K	18.22M	68.63M	2h09m37s	>300h	$>1.39\times10^2$	1h34m45s	7.31×10^{-3}	>300h	3m03s	2.35%
MySQL	1.61B	9.21M	859.56K	880.33M	1.15B	169h40m11s	136h38m23s	8.05×10^{-1}	54m13s	5.33×10^{-3}	>300h	31m57s	0.31%
Total	2.26B	9.24M	1.00M	916.33M	1.38B	174h20m56s	>899h48m28s	>5.16	3h06m05s	1.78×10^{-2}	>3000h	48m33s	0.46%

在10个中大型Linux程序上进行实验：

1. 并发漏洞**检测数量显著超过现有方法**，且保持**较低的检测耗时**。
2. 指令重排时进行指针流分析和修正的算法（ConPTA）仅占总时间开销的0.46%。

实验结果证明了本方法的有效性和高效性。

蔡彦
中国科学院软件研究所
yancai@ios.ac.cn 010-62661656

Y. Guo, S. Zhu, Y. Cai, L. He and J. Zhang
"Reorder Pointer Flow in Sound Concurrency Bug Prediction," 2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE), Lisbon, Portugal, 2024, pp. 203-215