中国科学院软件研究所学术年会’2020
暨计算机科学国家重点实验室开放周

面向二进制程序崩溃的多级释放后重用漏洞检测技术

和亮 苏璞睿 杨轶 闫佳 黄桦烽

联系方式: **heliang@iscas.ac.cn**

技术背景：

释放后重用漏洞是目前公认的最为常见也是可利用性最高的一类漏洞。现有检测方案的共同特点是需要从程序正常执行开始，如图1左侧，记录海量的内存分配、释放以及重分配等行为，导致检测与判定效率低下，同时由于存在如图2所示的多级释放后重用，使得现有检测方案难以准确检测出所有问题。

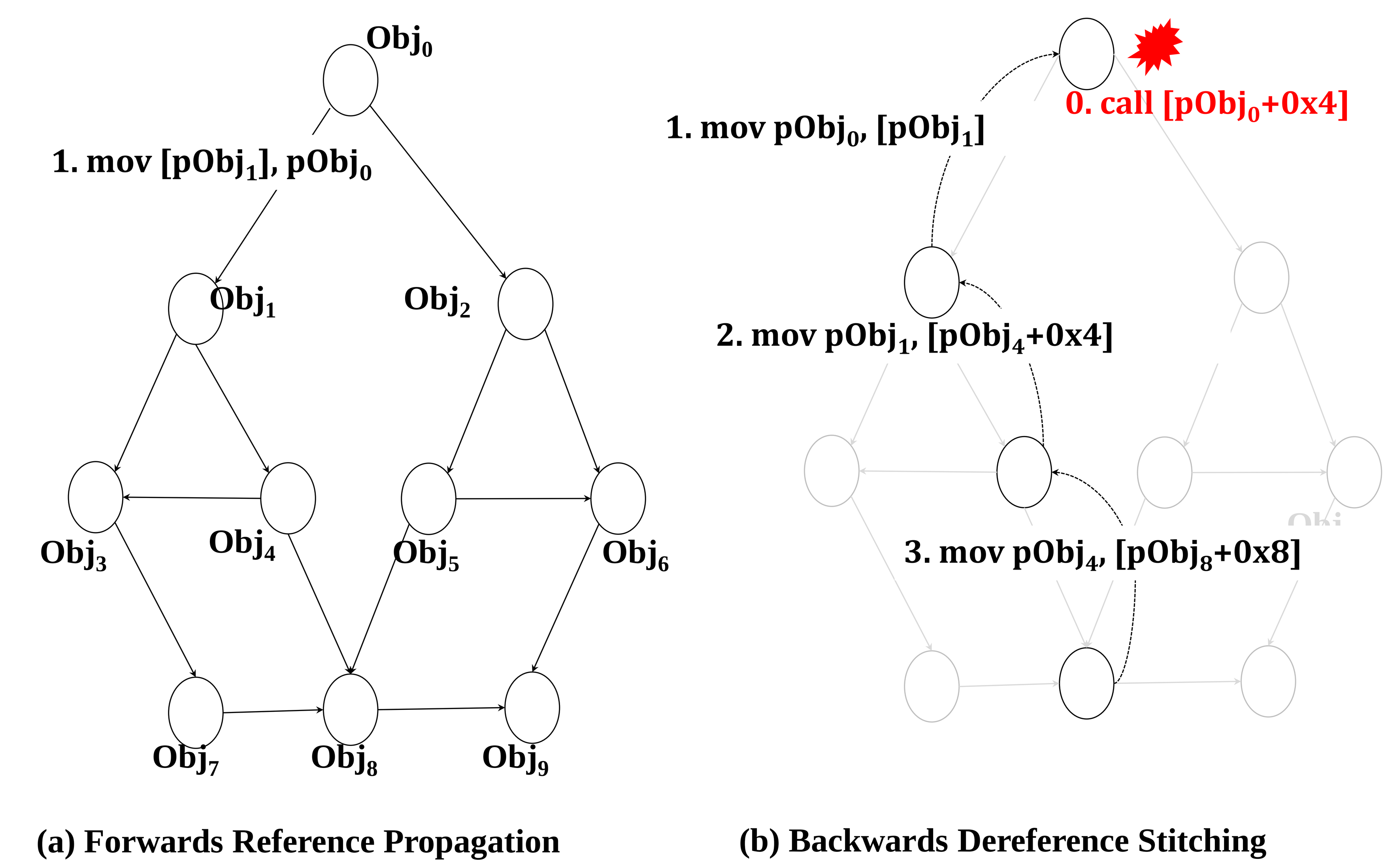

图1. 正向检测方案 vs. 回溯检测方案

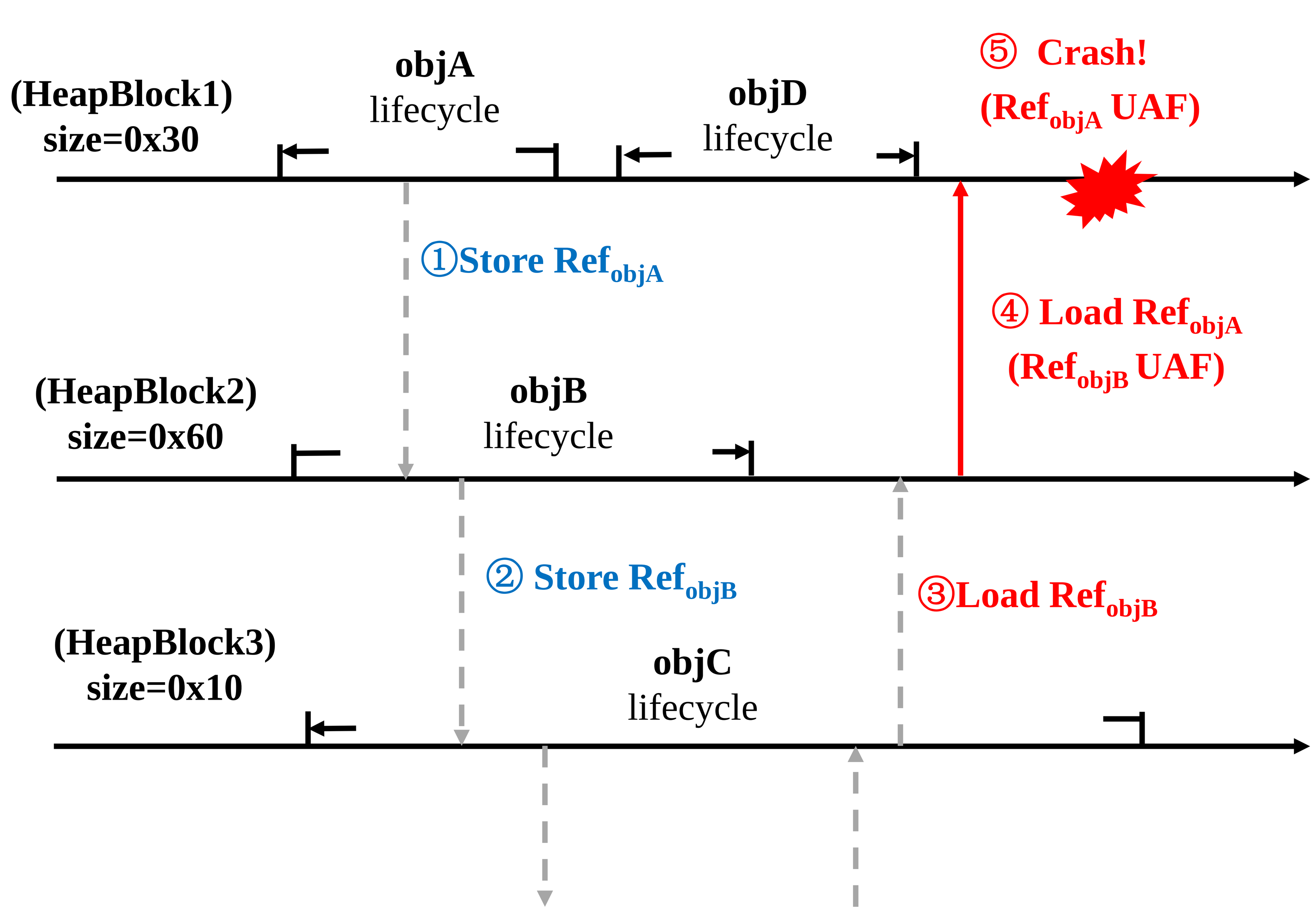

图2. 多级释放后重用导致的漏洞检测困难

技术思路：

如图1右侧，通过实时捕获程序运行时的内存异常访问所导致的崩溃，随后借助多级逆向切片技术来实现多级指针解引用的恢复，最后通过对每一级的目标对象实施“最后一次赋值”时刻查询，并与目标对象的内存分配与释放时间区间进行关联分析，从而不仅能够快速判定与检测出释放后重用漏洞发生的位置，同时还能有效检测出多级释放后重用的位置。

数据集：

Prog.	BugId	Version	Patch Info	Symbol Needed
IE	2010-0248	8.0	KB978207	
	2010-0249	8.0	KB978207	
	2010-3971	8.0	KB2482017	
	2011-1260	8.0	KB2530548	ReadFile
	2012-1875	8.0	KB2699988	(kernel32.pdb)
	2012-4787	9.0	KB2761465	
	2012-4792	8.0	KB2799329	RtlAllocateHeap
	2012-4969	8.0	KB2744842	RtlReAllocateHeap
	2013-0025	8.0	KB2792100	RtlFreeHeap
	2013-1306	9.0	KB2829530	KiUserException
	2013-1347	8.0	KB2847204	(ntdll.pdb)
	2013-3163	8.0	KB2846071	
	2013-3893	8.0	KB2879017	CTreeNode::
	2013-3897	8.0	KB2879017	AddRef
	2014-0282	8.0	KB2969262	Release
	2014-1776	8.0	KB2965111	CBase::
	2014-1815	8.0	KB2962482	PrivateAddRef
	2015-2425	11.0	KB3076321	PrivateRelease
	2017-11810	11.0	KB4040685	(mshtml.pdb)
	2018-8174	11.0	KB4103712	

Prog.	BugId	Version	Patch Info	Symbol Needed
PHP	2014-8142	5.4.34	5.4.35	emalloc efree
	2015-0231	5.4.36	5.4.37	zval_addref_p zval_delref_p
libkern	2016-1828	2782.40	3247.1	(k)malloc (k)free
	2016-4656	3248.60	3789.21	OSObject::retain OSObject::release
Python	**2019:**			
	38588-01	3.8	2ee879	
	38588-02	3.8	2ee879	
	38588-03	3.8	2ee879	pymalloc_alloc
	38610-01	3.8	296d45	pymalloc_free
	38610-02	3.8	296d45	_Py_INCREF
	38610-03	3.8	296d45	_Py_DECREF
	2020:			
	39421-01	3.8	993811	
	39453-01	3.8	3c57ca	

检测效果：

Prog.	Crashes	Crash Reproduction		UAF Ident.	
		# of Inst.	Crash Point	Crash Object	Ref Offset
IE	2010-0248	7,427,149	push [eax+0x2c]	UAF	0x10
	2010-0249	12,474,361	mov eax, [ecx]	UAF	x40
	2010-3971	18,694,213	cmp [ecx+0x18], 0x1	UAF	–
	2011-1260	11,522,711	mov edx, [eax+0x70]	**MUAF**	0x40
	2012-1875	17,823,049	call edx	UAF	0x4
	2012-4787	6,332,201	call [ecx+0x18]	UAF	0x4
	2012-4792	15,641,500	call [eax+0xdc]	UAF	0x4
	2012-4969	15,343,309	mov eax, [edi]	UAF	0x4
	2013-0025	12,755,301	mov edx, [eax+0x70]	**MUAF**	0x40
	2013-1306	84,579,652	call eax	UAF	0x4
	2013-1347	7,436,058	call edx	UAF	0x4
	2013-3163	13,869,856	call edx	UAF	0x4
	2013-3893	11,305,996	call [ecx]	**MUAF**	0x40
	2013-3897	11,941,540	call [eax]	UAF	0x4
	2014-0282	8,294,655	call [eax+0x1c8]	UAF	0x4
	2014-1776	29,679,527	cmp [ecx+0x184], 0x0	**MUAF**	0x4
	2014-1815	11,902,417	call [eax+0x9c]	UAF	0x4
	2015-2425	32,374,980	jmp eax	–	–
	2017-11810	14,820,690	mov al, [esi+0x3]	–	–
	2018-8174	17,597,824	call [ecx+0x8]	UAF	0x4

Prog.	Crashes	Crash Reproduction		UAF Ident.	
		# of Inst.	Crash Point	Crash Object	Ref Offset
PHP	2014-8142	9,740,711	mov edx, [edi+0x8]	**FF**	0x8
	2015-0231	9,755,428	mov eax, [eax+0x48]	**FF**	0x8
libkern	2016-1828	2,705,357	mov eax, [eax]	UAF	0x4
	2016-4656	12,474,361	mov eax, [ecx]	UAF	0x4
Python	38588-01	70,549,942	mov eax, [eax+0x64]	UAF	0x10
	38588-02	70,562,493	test [ecx+0x75], 0x1	UAF	0x10
	38588-03	70,437,139	mov eax, [eax+0x64]	UAF	0x10
	38610-01	71,159,071	cmp [eax+0x44], 0x0	UAF	0x10
	38610-02	71,066,490	test [ecx+0x75], 0x2	UAF	0x10
	38610-03	70,921,473	cmp [eax+0x44], 0x0	UAF	0x10
	39421-01	70,437,139	test [ecx+0x75], 0x1	UAF	0x10
	39453-01	70,645,124	mov eax, [eax+0x64]	UAF	0x10