FreeWill: Automatically Diagnosing Use-after-free Bugs via Reference Miscounting Detection on Binaries

和亮，胡宏，苏璞睿✉，蔡彦，梁振凯

USENIX Security 2022

heliang@iscas.ac.cn, purui@iscas.ac.cn

研究背景：


释放后重用漏洞（Use-after-free，简称UAF）是目前可利用性最高的一类内存破坏型漏洞。近年来，通过该类型漏洞造成了多起互联网范围内的网络攻击：例如，感染全球上百万Windows桌面系统的BlueKeep蠕虫攻击，正是利用RDP协议实现中的UAF漏洞。

```
1.  objP = malloc(OBJECT);
2.  objP->age = 1;
3.  strcpy(objP->name, “hex”);
4.  ……
5.  free(objP)
6.  ……
7.  printf(“%s\n”, objP->name);
```

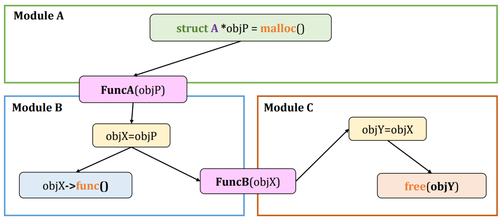
典型UAF漏洞示例及说明

漏洞基本原理：

程序在释放一块堆内存后（左侧第5行），再次解引用被释放对象指针（左侧第7行），导致程序进入不可预期状态（例如，崩溃）。

UAF漏洞机理分析及修复挑战：

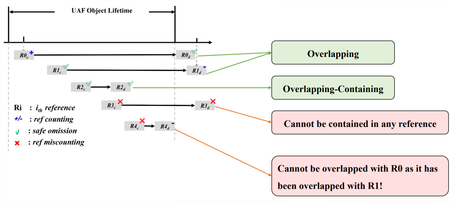
现有安全研究人员主要关心的是如何高效去发现新的UAF漏洞，例如UAFuzz、UAFL等工作。但随着该类型漏洞的不断发现，人们意识到如何分析和修复这些UAF漏洞是目前最为关键的问题之一，因为该类型漏洞往往需要跨函数、甚至是跨模块的方式进行根因分析之后，才能最终给出合理的修复方案。



真实场景下由跨模块指针传播导致的UAF漏洞

案例观察及引用计数模型：

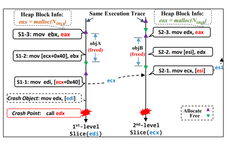
通过大量漏洞案例分析发现，很多UAF漏洞是由于引用计数缺失导致。即，程序使用引用计数进行堆内存管理，当有新的指针产生时，增加引用计数，当指针被清空时，减少引用计数，当引用计数清零时，立即释放该对象。但当人们过度优化引用计数操作时，导致对象提前释放，进而引发UAF漏洞。

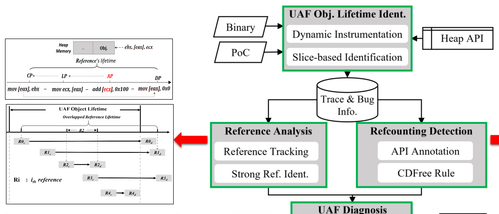


由引用计数缺失导致的UAF漏洞模型

UAF对象生命周期识别：

通过面向堆指针的污点传播方法，设计并实现了多级指针逆向切片的UAF根对象细粒度跟踪方法，可以有效识别二进制程序中的UAF对象及其完整生命周期





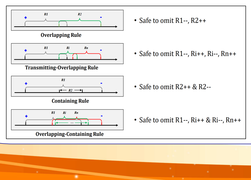
FreeWill: UAF漏洞根因推理及修复方案

UAF对象引用识别：

通过识别对象堆内存指针、栈内存指针以及寄存器指针，并根据其生命周期长短进行“强引用”指针筛选，随后收集每个该类型指针并构建相应的生命周期模型。

引用计数关联：

通过启发式规则实现二进制对象的引用计数行为识别与提取，并在此基础上基于引用创建和销毁行为与引用计数行为的执行距离，实现两者的语义关联分析



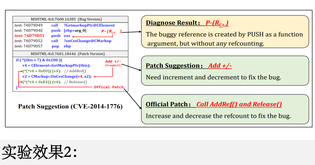
基于引用计数优化模型的UAF根因推理：

通过提出四个引用计数优化模型，并借助上述分析得到的UAF对象生命周期内的各引用及其关联的计数操作，设计并实现UAF漏洞根因推理算法

Dataset	Web Browser (32)			Kernel (21)		Script Engine (23)	
	IE	Chrome	Firefox	Linux	MacOS	Python	PHP
Bug Num.	20	7	5	15	6	15	8
no INC, no DEC	14	0	0	6	2	10	4
no INC, has DEC	0	0	0	6	3	2	1
Dangling Usage	4	6	4	2	1	0	2
NULL-deref	0	0	1	1	0	3	0

实验效果1：

针对76个大型二进制软件中的UAF漏洞，可成功提取所有UAF对象，并对其中66个漏洞完成根因分析，其中56个漏洞修复建议和官方补丁完全一致



实验效果2：

通过细粒度分析所有漏洞的引用计数行为，我们可以根据缺失引用计数的程序位置，自动给出准确的补丁修复建议，并和官方补丁位置一致

实验效果3：

将本文所提出方法应用在最新版Linux内核中，发现351处引用计数错误，其中240个补丁被合并至主线版本