# 取消悬挂：早期检测悬空指针释放重用和双重释放漏洞

## 摘要

Use-after-free漏洞正在迅速普及，特别是用于利用Web浏览器。释放后使用（和双重释放）漏洞是由在悬空指针上运行的程序引起的。在这项工作中，我们建议及早发现，一种新颖的运行时方法，用于发现和诊断释放重用和双重释放漏洞。先前的工作主要集中在制造漏洞（即使用悬空指针），早期检测将重点转移到漏洞根源处的悬空指针的创建。

早期发现可通过识别以下内容来提高测试的有效性：程序执行过程中不安全的悬空指针创建但未被使用的地方。通过自动收集有关该漏洞涉及的所有悬空指针的信息，它还可以加速漏洞分析并最大程度地减少未完成修复的风险。我们在名为Undangle的工具中实施了我们的早期检测技术。我们在8个现实漏洞上进行漏洞分析来评估Undangle。分析发现，Firefox中的两个独立漏洞具有共同的根本原因，并且它们的补丁程序不能完全修复潜在的错误。我们还评估了Undangle，以便在Firefox Web浏览器上进行测试以识别潜在的漏洞。

## 1.引言

当指针指向的对象被释放时，将创建一个悬空指针，留下指向死内存的指针，该内存以后可以重新分配或覆盖。悬空指针严重影响程序的正确性和安全性，因为它们会向释放重用和双重释放漏洞打开大门，这两个重要的漏洞类别是程序通过悬空指针在内存上运行产生的。

释放重用和双重释放漏洞是可利用的[7，22]，并且与其他漏洞（如缓冲区和整数溢出）一样危险。释放重用漏洞特别隐蔽：它们已被用来发起许多零日攻击，包括对Google和Adobe公司网络的Aurora攻击[44]，以及在过去一年中对Internet Explorer的3次零日攻击[3,24,41]。

我们对CVE数据库[4]中公开披露的释放重用和双重释放漏洞的分析揭示了两个令人不安的趋势。如图1所示：（1）UAF漏洞的流行度迅速增长，自2008年以来，每年的数量翻了一番以上（同期，每年报告的漏洞总数实际上正在减少），（2）使用复杂数据结构并需要手动内存管理的语言（例如C / C ++）编写的Web浏览器（69％）和操作系统（21％）上存在大量的After-after-free和double-free漏洞。

Use-After-Free和Double-free漏洞很难识别并且要花费大量时间来诊断，因为它们涉及两个单独的程序事件，这些事件可能在时间上相隔很远：悬挂指针的创建和使用（取消引用或Double-free）。 此外，了解根本原因可能需要对内存中的多个对象进行推理。尽管有些悬空指针是通过忘记置释放对象的指针无效（非共享错误）而创建的，但其他悬空指针则涉及多个对象，这些对象共享一个被释放的对象（共享错误）。

发生共享错误是因为并非所有父对象都知道子对象释放。对于组件使用不同内存管理方法构建的Web浏览器而言，它们尤其成问题。例如，在Firefox中，对JavaScript对象进行垃圾回收，对XPCOM对象进行引用计数，并且布局引擎使用手动管理。这种混合使得使用不同的内存管理方法很难推断出代码之间共享的对象，这尤其容易导致指针错误。