# 背景介绍

软件漏洞是攻击计算机系统最常见的媒介。尽管人们在设计、构建和部署没有缺陷的软件方面付出了很大的努力，但即使是相对简单的软件系统中也仍然有漏洞存在。更加严重的是，即使在可信计算基中也可能存在足以令攻击者绕开系统中的安全机制的严重漏洞。

在所有的软件漏洞中，内存破坏型的漏洞是最常出现，也是最严重的漏洞之一。这有以下几个原因：首先，大量的程序是由C或C++这种类型不安全的语言编写的，这使得内存破坏型漏洞极为常见；同时，内存破坏漏洞的攻击能力非常强，很多攻击形式都利用内存破坏漏洞来完成攻击，可以导致的结果下至加密密钥口令等关键数据的泄露，上至系统最高权限被攻击者夺取。由于这些原因，内存破坏型的漏洞今天仍然是最常被利用的漏洞之一。根据微软公司2015年的报告[1]，基于内存破坏的攻击方式占据了远程代码执行类CVE（Common Vulnerabilities and Exposures）的主导地位。

鉴于内存破坏攻击的严重性与广泛性，研究者在如何防范这类攻击上进行了大量的工作。一般来说，现有的防御措施可以分为两类：内存错误的检测和特定攻击形式的缓解对策。

第一类防御措施致力于检测内存破坏攻击成立的源头。由于内存破坏攻击需要依靠内存错误来触发，因此如果能够阻止内存错误的发生，那么就可以从根源上阻止内存破坏公积。但是为了根除内存错误，需要在软硬件上付出比较很大的额外开销，根据Szekeres等人的总结[2]，这类措施带来的额外开销可以达到50%甚至超过100%。而当设计者需要在性能与安全之间进行取舍时，往往是安全特性要为性能让位，因此大多数的内存错误检测措施只用在离线的漏洞检测或者崩溃分析中。

第二类防御措施则为特定形式的攻击提供针对性的防御措施，致力于阻止攻击者实行有害系统安全的行为。内存破坏攻击的形式可以分为四类：代码破坏、控制流劫持、数据导向攻击和信息泄露攻击。对于每一种攻击形式，研究者们开发了有针对性的防御手段。例如，数据执行保护（Data Execution Prevention，DEP）机制要求一块内存区域不得同时具有写权限和执行权限，从而阻止控制流劫持中的代码注入攻击。针对性的防护措施具有额外开销小的优点，通常这类措施只会带来大约10%或更低的额外开销。但是针对性措施也有其缺点，那就是只能防止一种类型的攻击，这就导致这类防御措施可以被绕开。

总的来说，内存破坏型攻击的防御措施有这样的特点：能够提供强有力安全保障的解决方案会拖慢系统速度，而效率更高的解决方案则只能提供有限的防护。

# 攻击类型

为了应对内存破坏攻击，我们首先需要了解内存破坏攻击是如何引发并利用内存错误的。下图是内存破坏攻击的一般攻击模型，其中每个矩形框中表示的是攻击中的一步，最下面一行的椭圆框中表示的是具体的攻击种类，菱形框则表示在具体攻击方式中的选择。



图 1 内存破坏攻击的攻击模型

## 制造内存错误

内存破坏攻击首先需要制造内存错误，即图1中的最初两步。第一步中攻击者将制造一个无效的指针，第二步中攻击者将使用这个指针进行读写，即“解引用”（dereference），从而触发一个内存错误。制造无效指针的方法有两大类，一类是从空间上令指针无效，也就是制造一个越界的指针；另一类是让指针在时间上无效，例如解引用一个指向被删除对象的指针，这样的指针称为悬垂指针。解引用越界指针所引发的内存错误称为“空间错误”，而由悬垂指针引发的内存错误则称为“时间错误”。

攻击者可以利用多种程序漏洞来制造越界指针。例如，攻击者可以触发一次指针的分配失败，如果这个失败事件因为程序漏洞而没有受到检查，那么攻击者就得到了一个空指针，并可以进一步利用这个空指针在内核空间中展开攻击{Kemerlis, 2012 #158}。另一种常见的制造越界指针的方法是在循环中不断地增加或减小数组指针的值，如果程序中没有进行边界检查，或者边界检查不完整，那么攻击者就得到了一个超出数组边界的指针，这种方法称为缓冲区上溢出和缓冲区下溢出。

# 参考文献

[1] RAINS T, MILLER M, WESTON D. Exploitation trends: From potential risk to actual risk; proceedings of the RSA Conference, F, 2015 [C].

[2] SZEKERES L, PAYER M, WEI T, et al. SoK: Eternal War in Memory; proceedings of the 2013 IEEE Symposium on Security and Privacy, F 19-22 May 2013, 2013 [C].