

home (/) edit (/new/send) share (/wp-log-in.php?action=logout&redirect_to=http%3A%2F%2Fdrops.wooyun.org)

Windows平台内存防护与绕过技术的进化演变系列之（一）内存攻防发展概述

Chuck (/author/Chuck) · 2015/05/12 10:32



(/author/Chuck)

Chuck (/author/Chuck)

0x00 引言

最初有总结下的想法始于近期看过的几篇关于CFG绕过以及EMET绕过的文章，这些文章里大概都有提到类似这样的句子“*As with every known exploitation mitigation, there are ways to bypass it if certain conditions are met.*”。无论攻与防的技术都是有时间与条件限定的，某种技术一般只在某段时间内对某特定场景有效，而攻与防又是在互为前提、互为基础、又互为对手彼此砥砺发展的，那么如果对这些技术的由来以及互相之间的争斗历史（历程）有一个宏观整体的、脉络清晰的认识，对于学习理解Windows平台内存防护机制以及新出现的各种内存攻防技术应该是有积极意义的。

其实很早之前就已经有人做过类似的总结，早期国内的nsfocus、xfocus、邪恶八进制，国外的微软安全部门以及一些早期黑客组织的牛们都写过关于内存攻击防护、缓冲区溢出相关的总结，内容侧重点各有不同，有讲漏洞原理分析的、有漏洞利用的、有侧重研究防护与缓解技术的，本系列拟在前人总结的基础上，结合近几年新出现的一些思路、技术、手段，把各种防护与绕过技术从头至尾连接起来，尝试做一个平衡、全面的讲述。在写的过程中，会尽量选择添加一些实例分析来增加文章的可读性，但水平所限，难免顾此失彼或出现各种错漏，恳请各位大神谅解并多批评指正。

0x01 文章结构

0x02对内存缓冲区溢出进行概述，主要分三块内容：什么是缓冲区溢出、缓冲区溢出原因、早期缓冲区漏洞利用技术发展简史。0x03讨论内存防护机制的出现，主要分也分三块内容：内存防护的安全需求、防护原则、防护策略。0x04简单罗列目前主流的攻防技术。

0x02 内存缓冲区溢出概述

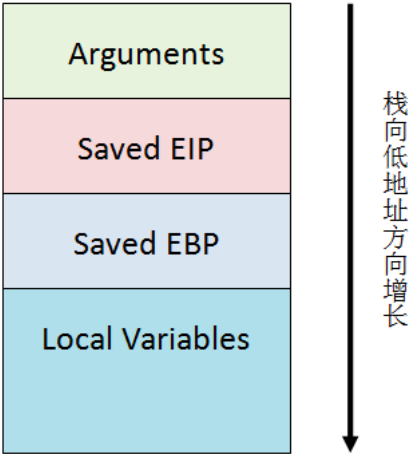
内存安全违规类漏洞可以泛指一切访问内存时引入的安全缺陷利用，如缓冲区溢出，Use-After-Free、Double Free等，其中缓冲区溢出攻击是最基础最典型也可能是出现最早的内存类安全问题。内存攻击利用的历史也是从缓冲区溢出攻击开始的。（本章纯基础，神请绕过）

1、什么是缓冲区溢出

计算机程序一般都会使用到一些内存，这些内存或是程序内部使用，或是存放用户的输入数据，这样的内存一般称作缓冲区。缓冲区溢出指的是计算机没有对接收的输入数据进行有效的长度检测，输入数据的长度超过了程序缓冲区本身的容量，而导致数据溢出到被分配空间之外的内存空间，使得溢出的数据覆盖了其他的内存空间的数据。

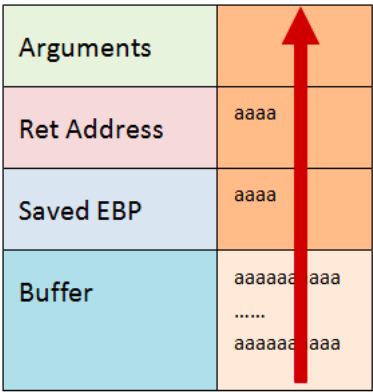
简单来说就是用户输入的东西太多，覆盖了不该覆盖的地方，从而使程序执行跳转到不该跳转的地方。

x86的栈结构：



```
void overflow_basic()
{
    char buffer[30];
    gets(buffer);
    printf("%s\n",buffer);
}
int main()
{
    overflow_basic();
    return 0;
}
1
```

当输入超过30个'a'字符时，会覆盖掉前栈帧的EBP和函数返回地址，如下图所示：



2、缓冲区溢出原因

缓冲区溢出攻击出现的根本原因源于现代计算机的基础架构。现代计算机指的是冯诺依曼体系架构的计算机，在冯氏体系计算机中，程序和数据是以二进制代码的形式不加区分地存放在存储器中的，存放位置由地址决定。既然程序与数据在内存中的存放形式是没有区别的，那么用户输入的数据在理论上当然也是可以作为程序执行的。缓冲区溢出发生后，数据覆盖改变了程序的执行流程，使程序能够跳转到用户构造的数据区执行用户指定的代码，攻击就发生了。

缓冲区漏洞的出现与C语言的出现与流行是脱不开关系的，C/C++支持内存的分配与回收，以及指针的计算、转换等，语言本身也没有对数组边界进行检查等内存保护，这些都为缓冲区溢出漏洞的出现以及攻击利用提供了条件。

3、早期缓冲区溢出利用技术发展简史（1988-2004）

上世纪70年代末C语言出现并逐渐流行，到了80年代国外的黑客就开始意识到缓冲区溢出的问题并开始加以利用。较为典型的或有里程碑意义的事件如下：

- 早期最著名的一次（也许是第一次）缓冲区溢出攻击是1988年莫里斯蠕虫病毒。作者Robert T. Morris利用Unix fingerd服务程序没有限制输入长度的漏洞，输入512个字符后导致缓冲区溢出。

随后第二年（1989年）Spafford提交了一份VAX机上的BSD版UNIX的fingerd缓冲区溢出程序细节的技术报告，开始引起一部分安全人士对缓冲区溢出领域研究的重视。之后又有L0pht heavy Industries的Mudge写了一篇利用BSDI上的libc/syslog缓冲区溢出漏洞的文章。但并未引起广泛的关注。

注：莫里斯蠕虫病毒造成了9600万美元的损失。

- 1996年，出现了具有里程碑意义的一篇文章---|||Aleph One在Underground发表了《Smashing The Stack For Fun And Profit》。这篇文章详细描述了linux系统栈的结构，以及如何利用基于栈的缓冲区溢出。受这篇文章启发，讲述如何利用缓冲区溢出并写一段所需的shellcode的文章开始在网络上大量涌现。

国外是这样评论Aleph One这篇文章的：“Everything started with Aleph One's paper”。Aleph One前辈另一处值得我们记住的是，他是shellcode一词的发明者。Aleph One文章中最初指的是如何在linux上写一个开shell的Exploit。

- 1997年，Smith在以前文章的基础上，收集了各种处理器体系下的Shellcode，提出了在各种Unix变种中写缓冲区Exploit更详细的指导原则。

Smith在文章中还谈到了类Unix系统的一些安全属性，并对安全编程进行了讨论。

- 1998年，国外黑客团队“死牛崇拜”（Cult of the Dead Cow）的Dildog在Bugtrq Maillist中以Microsoft Netmeeting服务缓冲区溢出为例，详细介绍了如何利用Windows平台的栈缓冲区溢出漏洞。可以说Dildog为Windows下缓冲区漏洞利用奠定了基础。

这篇文章提出了利用栈指针的方法来完成跳转，Windows下的溢出利用迈出了关键一步。Dildog还有另一篇可与Aleph One的《Smashing The Stack For Fun And Profit》齐名的经典之作《The Tao of Windows Buffer Overflows》（Windows缓冲区利用之道），着重讲述了Windows平台下的利用技巧。尽管对于今天来说文中的技术已老去而不再适用，作者文中的一些想法以及思考问题的角度直到今天还是值得我们借鉴的。（例如利用程序中已有的代码，如call *||或jmp *||来让程序流转向shellcode执行；还有小的shellcode与达成实际利用的exp程序分开；加壳消除shellcode中的空字符等等）

- 1999年也是Windows缓冲区漏洞利用技术发展的很重要的一年，这一年中发生了这么几件事：一件是Dark Spyrit在Phrack 55上提出使用系统核心DLL中的Jmp ESP指令完成跳转(如上文所述，这种想法其实Dildog之前也提起过)，推动Windows溢出利用迈出实质性的一步；另一件是Litchfield详细讨论了Windows NT的进程内存以及栈结构，以及基于栈的缓冲区溢出，并以rasman.exe为对象，给出了提权创建一个本地shell的shellcode；还有一件是w00w00的M. Conover写的基于堆的缓冲区溢出教程。他在研究中注意到当时的内存保护方法如非执行栈等是无法防止基于堆的溢出的。

可以说，2000年之前，缓冲区溢出漏洞利用作为漏洞利用中一种“杀伤性武器”所应具备的所有条件都已经具备了，剩下的只是一个契机来爆发。而2000之后个人电脑与Windows系列操作系统以及互联网的推广应用都是其爆发的契机。让我们看一下接下来的发展（2001年至2004年缓冲区溢出攻击发展达到一个高峰，在所有CVE漏洞排名里连续四年排名第一）：

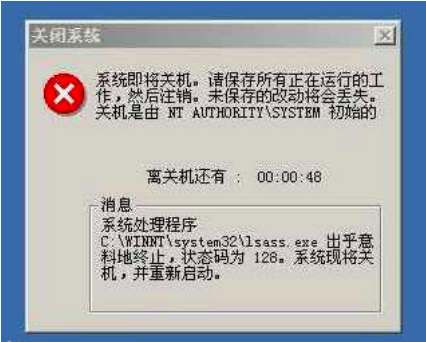
- 2000年，Cerberus小组发布了微软IIS 4/5的一个缓冲区溢出漏洞，2001年开始，Microsoft的IIS5.0一系列的漏洞被发现，该系列漏洞很多都是由于Unicode字符集的处理问题造成的。因为IIS是Web服务器程序，因此，该漏洞给网站的安全构成了极大的威胁。

Code Red蠕虫就是一个应用IIS漏洞的一个典型例子，由于其主要针对英文版本的Windows NT/2000操作系统，因此国内使用中文版系统的大都没感觉到什么影响。Code Red病毒实际上造成了26亿美元的损失。



如果说大多数人对Code Red没什么印象的话，接下来的2003年的“冲击波”，一定让所有经历过的人都印象深刻。

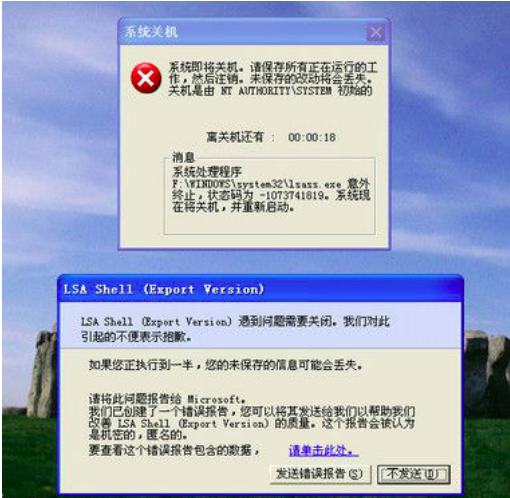
- 2003年，引起全球轰动的“冲击波”病毒及其变种，就是利用Windows RPC服务的缓冲区溢出漏洞（著名的MS03-026、MS03-039）来进行传播的。中毒后的症状是系统频繁重启，所有网络服务均出现故障，无法复制粘贴、文件显示不正常等等。



一个趣闻是，蠕虫“冲击波”变种病毒(Blaster.B)的作者，18岁的美国少年Jeffrey Lee Parson在程序中内嵌了自己的名字，FBI因此顺利找到并逮捕了他。

2003年另一个著名的病毒是Mydoom，这两种病毒共造成了数十亿美元的损失。

- 2004年，另一个让人印象深刻的“震荡波”病毒及其变种爆发，同样也是利用了Windows RPC的缓冲区溢出漏洞（MS04-011）。



时间大概以2004年为分界线，如果说2004年之前，缓冲区漏洞进行攻击是一种非常方便又简单易用的漏洞利用形式的话（Crispin Cowan在《Buffer Overflows:Attacks and Defenses for the Vulnerability of the Decade》中这么总结的“Buffer overflow attacks form a substantial portion of all security attacks simply because buffer overflow vulnerabilities are so common [15] and so easy to exploit [30, 28, 35, 20]. ”），那么2004年之后，这种情况就开始发生了变化。04年之后，新版本类Unix和Windows操作系统应用得越来越广泛，新版本操作系统中都引入了一些内存防护机制，为内存攻防战场增加了新的变数。

有一些防护技术，如/GS、SafeSEH等是在04年之前出现的，尽管技术出现了，大多数人在使用的还是老版本操作系统，因而类似GS的防护技术实际上在用户计算机上并没有得到广泛应用。

0x03 内存防护机制的出现

1、安全需求

前文我们已经提到，缓冲区溢出漏洞可以造成严重的后果，在较为严重的情况下，软件漏洞可以使攻击者获取用户计算机的完全控制权限，从而在用户机上为所欲为。在早期，通用的应对漏洞的方法是为受感染的软件安装软件供应商提供的安全补丁。然而补丁更新的方法有着很明显的不足之处：

- 时间耗费大。在了解了漏洞的存在并熟悉了漏洞相关的情况后，软件供应商必须开发一个健壮的安全更新程序，该更新还必须要经过充分的测试，以保障其不会引入新的安全问题。从发现漏洞到补丁开发完成通常需要耗费大量时间，而用户在没有安装补丁之前将一直处于易受攻击的状态。

- 未知漏洞的存在。要想给漏洞打补丁，软件供应商必须具备该漏洞相关的先验知识，而软件漏洞的存在是不可避免的，在实际中，一定是有一些软件供应商并未发现的漏洞存在。对于这部分漏洞，当然也无法通过打补丁的形式进行防护。

2、防护原则

微软对于漏洞的防护原则大致有两个：要么阻止，使攻击者不能；要么提高其攻击成本，使其不愿。

基于这两种原则，在如何保护用户免受未知的或尚未解决的安全漏洞的危害，或在无法阻止的情况下，有效降低软件漏洞带来的安全风险方面，微软和其他的软件供应商付出了大量的努力。例如通过防火墙阻断连接、使用授权/验证技术阻止访问、关闭或停止有漏洞的服务等等。这些方法的目的只有一个：使攻击者成功利用漏洞变得很难或不可能。

而另一种可以在补丁未出来之前保护用户安全的研究方向就是我们所要讨论的内存防护技术。内存防护技术可以有效打断攻击者的攻击链条，使攻击者对内存的利用变得更加艰难，提高攻击的门槛，从而保护用户安全。

3、内存防护策略

- 增强不变量策略

一种可用以打破攻击技术的策略是通过引入新的不变量，以使攻击者攻击时对内存的隐含假设条件不再合法。采用这种策略的技术有DEP、SEHOP等。

- 不确定性策略

攻击者攻击时通常会假设攻击相关前提（模块地址、内存分布等）是确定的，通过增加系统的不确定性可以使攻击者的假设落空，从而阻断攻击程序的可靠运行。采用这种策略的技术有ASLR等等。

- 不可预测性策略

在一些场景下，通过利用攻击者不知道或不可简单预测到的信息，可以阻断攻击程序的利用过程。典型采用这种策略的技术有/GS等。

0x04 攻防技术概览

基于0x03章讨论的防护策略，在与攻击技术的对抗中，各种各样的防护技术开始发展并成熟起来。

本系列接下来将讨论的内存防护缓解技术如下表所示：

/GS Cookie
增强型/GS Cookie
SafeSEH
Safe Unlinking
堆元数据 Cookie
堆元数据加密
Dynamic SafeSEH
Hardware DEP（NX）
Permanent flag
ASLR、KASLR
SEHOP
CFG
EMET4.0-EMET5.2（防护技术集成）

相应的攻击利用以及防护绕过技术如下表所示：

基于栈的缓冲区溢出
GS Cookie 绕过（猜测/计算 Cookie、覆盖 SEH、覆盖虚函数指针）
SafeSEH 绕过（利用堆地址覆盖 SEH、利用未开启 SEH 保护的模块）
Safe Unlinking 绕过（lookaside list、堆喷）
Heap Cookie 及加密的绕过（猜测/计算 Cookie、堆喷射）
DEP 绕过（Ret2libc、TEB 过 DEP、NtSetinformationProcess、SetProcessDEPPolicy、ROP、SHE 绕过）
ASLR 绕过（覆盖部分返回地址、寻找未启用 ASLR 的模块、内存信息泄露利用、堆喷、Systemcall）
SEHOP 绕过（伪造 SHE 链）
CFG 绕过（未开启 CFG 的模块、覆盖返回地址、寻找未被 CFG 保护的直接调用）
KASLR 绕过（内核堆喷、利用非分页内存等）

0x05 内容预告

下一章将进行Windows XP下内存攻防的讨论（GS、SafeSEH、SEHOP、NX/W^X/DEP）。包括背景知识（Windows内存堆栈结构）、相关防护技术的引入与绕过、防护与绕过实例等等。

☆收藏 分享

H4TS

写下你的评论...

发表

- 

asnine 2015-11-03 14:24:53

楼主加油，继续写，还等着看下文呢。。


- 

k0_pwn 2015-05-14 22:28:59

DEP，和ASLR应该是现在主流的两种内存防护机制，DEP是最有意思的！我常用的是DisableDEP和VirtualAlloc，感觉构造ROP的过程也是很好玩的，支持楼主


- 

天清地宁 2015-05-12 17:53:44

@Chuck 你想多了。因为我也不懂又刚好对挖洞洞有些兴趣。所以求慢更，求不断更。(笑)



- 

Chuck 2015-05-12 11:19:36

@天清地宁 感谢支持，其实真写起来才发现自己不懂的好多，所以写的过程本身也是学习的过程，我会慢慢写，用心写，请大牛多多指教！


- 

天清地宁 2015-05-12 11:00:18

 沙发，火钳刘明，已阅. 乌云貌似以WEB为主，出现Windows内存攻防感觉挺难得的。希望楼主慢慢写下去。坐等更新。

 回复

感谢知乎授权页面模版