



X'CON 2003

Windows 2003 堆溢出及其利用技术深入研究

作者: FlashSky

日期: 2003-12-26

- 感谢

安全焦点所有的成员与启明星辰积极防御实验室的同事

- 主讲目录

- WINDOWS堆结构简介
- WINDOWS堆溢出利用
- WINDOWS 2003堆溢出保护
- WINDOWS 2003堆溢出保护的弱点
- 思路的延续：弱点利用深入的讨论
- 思路的突变：更广泛的另类有效利用的手段
- 未来WINDOWS 2003堆溢出利用研究的方向

- **WINDOWS 堆结构简介 (1)**

- ü 前言

- WINDOWS 2003, 系统低层的安全性改进
 - 本文的目的

- ü 当前关于WINDOWS堆的研究

- WINDOWS 堆管理结构与管理特性的研究
 - WINDOWS 堆特性在堆溢出时的利用的研究
 - WINDOWS 2003的堆特性的研究

- **WINDOWS 堆结构简介 (2)**

- ü WINDOWS堆结构简介

- 堆与堆块

- 堆管理结构与堆块管理结构

- 堆的整体结构构架

堆的基地址

堆管理结构块 (SIZE: 0X640/0XC50)
堆段表块0 (SIZE: 0X40)
小堆块分配管理表结构 (SIZE: 0X1818), 可选
用户堆块数据区

一般紧接着堆管理结构块之后

- **WINDOWS 堆结构简介 (3)**

- ü WINDOWS堆块分配与释放的管理

- 大堆块对象与小堆块对象

- 空闲堆块双向链表头

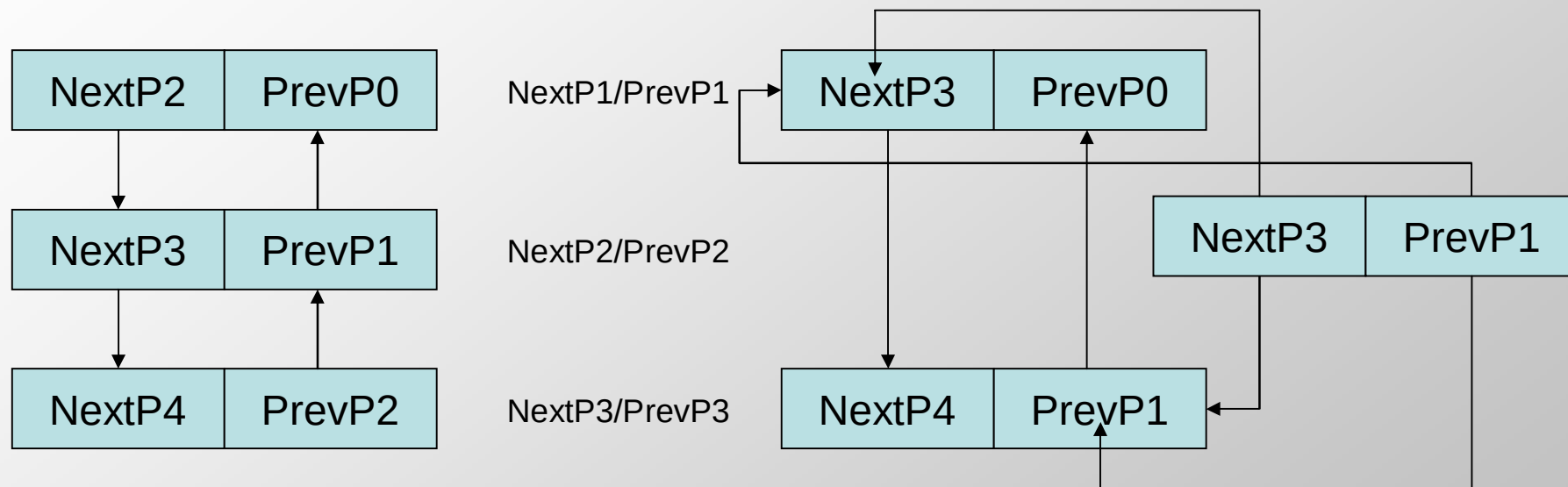
- 空闲大堆对象与空闲小堆对象入链算法

- **WINDOWS 堆溢出利用 (1)**

- ü 基本原理

- ü 常见的普通利用方式:

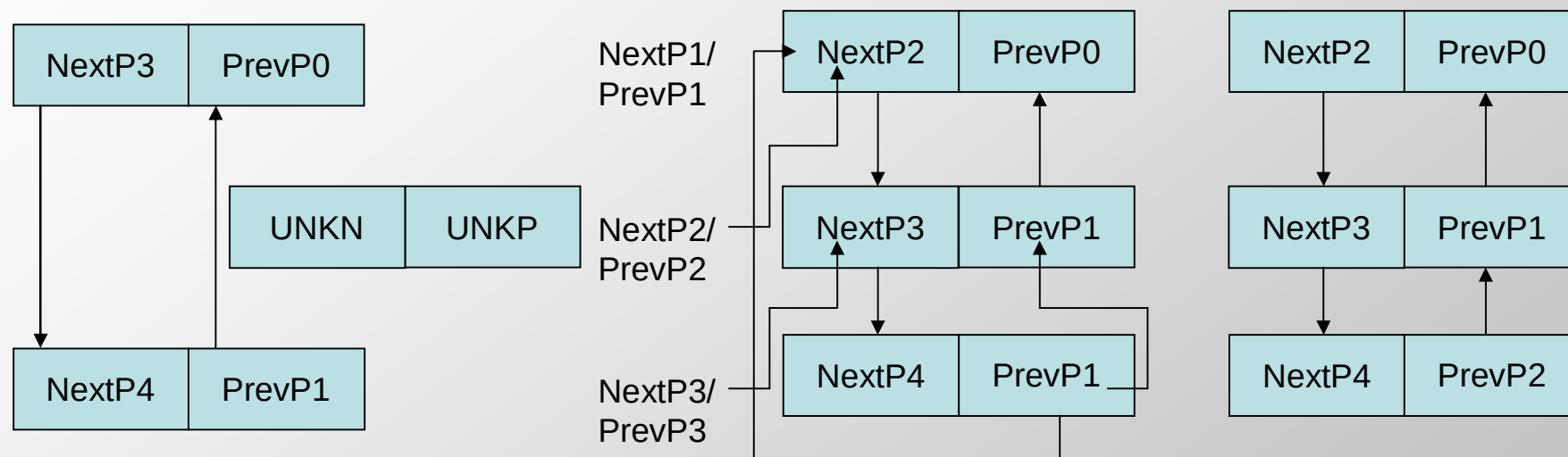
- 引起空闲堆组成的双向链表的脱链的操作



一个正常的堆块脱链表过程

- **WINDOWS 堆溢出利用 (2)**

- ü 双向链表的入链利用的可能性
 - 可利用的原理
 - 利用的要求

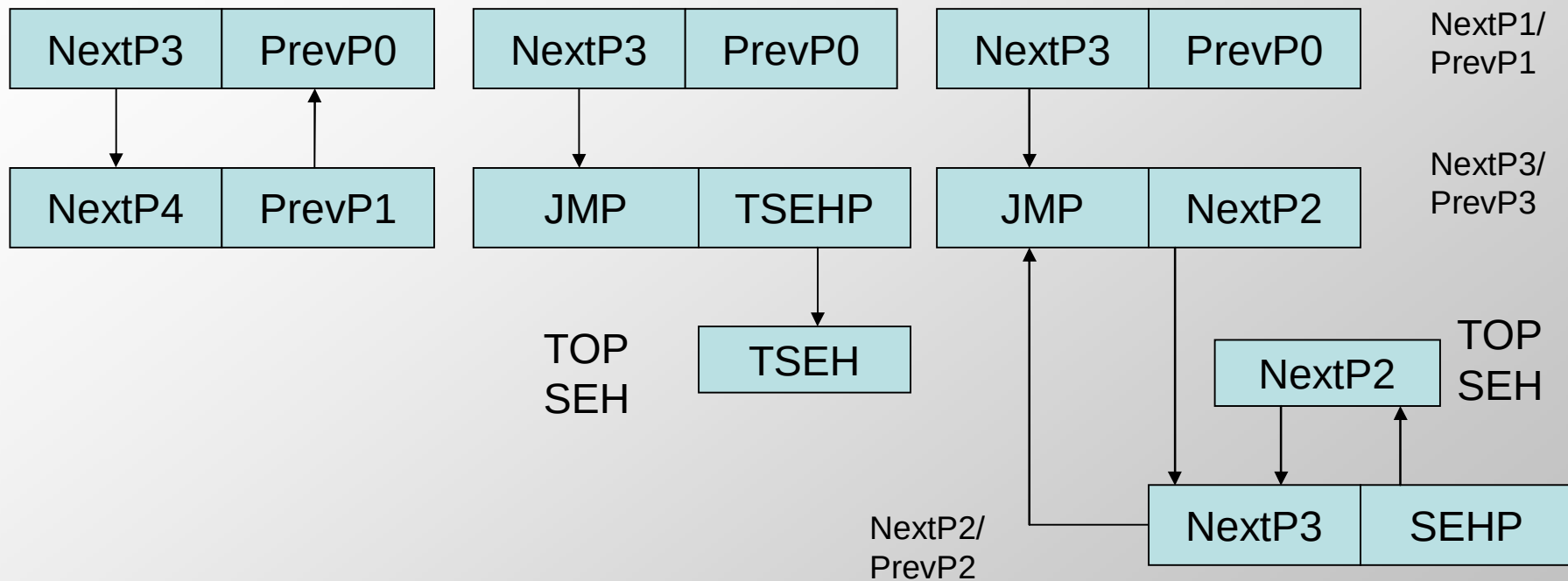


一个正常的大堆块插入入链表过程

- **WINDOWS 堆溢出利用 (3)**

- ü 覆盖空闲大堆可以达到的目的:

- 我们能将一个当前释放堆的地址填入到一个由我们指定的内存地址中



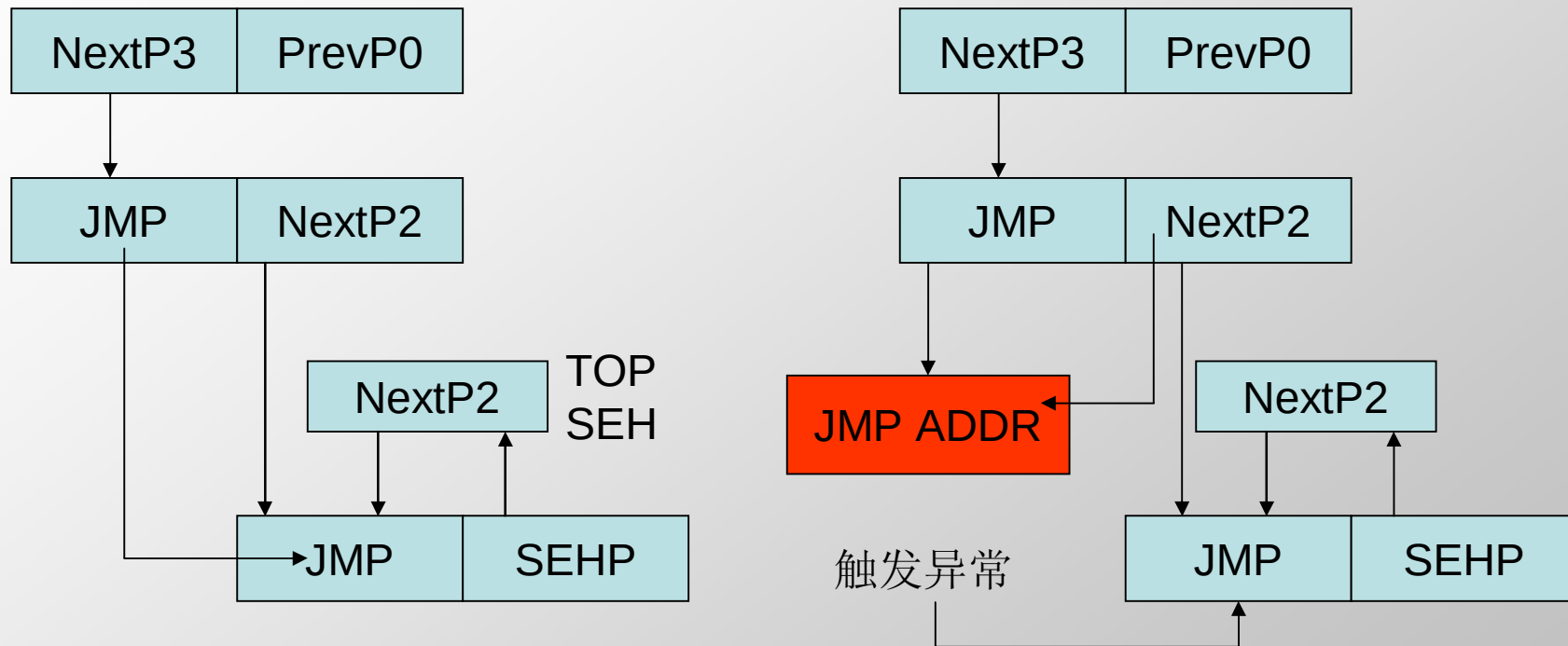
一个利用大堆块插入入链表的过程

- **WINDOWS 堆溢出利用（4）**

- ü 入链表利用的进一步

- 问题：当前释放堆块地址的前8字节内容是我们无法控制的指针价值。

- 解决方法：再次分配时候可以将我们可以控制的4字节写入



利用分配再次写入JMP代码到头四字节

- **WINDOWS 堆溢出利用（5）**
 - ü WINDOWS堆溢出可以利用的途径
 - 脱链表时
 - 入链表时

- **WINDOWS 2003 堆溢出保护 (1)**

- ü 基于堆溢出利用原理的检查

- 基于覆盖双向链表的地址导致可写入任一地址空间

- 检查上下链表的一致性可以检查和防止堆溢出的利用


```
lea    ecx, [esi+8]
mov     edi, [ecx]
mov     [ebp-0D8h], edi
mov     eax, [esi+0Ch]
mov     [ebp-94h], eax
mov     edx, [eax]
cmp     edx, [edi+4]
jnz     loc_77F36DE1
cmp     edx, ecx
jnz     loc_77F36DE1
mov     [eax], edi
mov     [edi+4], eax
```

WINDOWS 2003 的检查的代码

- **WINDOWS 2003 堆溢出保护 (2)**

ü 简单而言，其要求就是
被处理的当前堆块的管理结构的双向链表的下一个堆块指针与上一堆块指针要满足如下条件：

1. 当前堆块的下一个堆块指针指向的堆块的上一个堆块指针要等于当前堆块的地址
2. 当前堆块的上一个堆块指针指向的堆块的下一个堆块指针要等于当前堆块的地址

- **WINDOWS 2003 堆溢出保护 (3)**

- ü 普通绕过的困难

需要构造这样的条件需要准确的知道当前堆块的地址，然而往往当前堆块的地址我们不可知。

同时我们需要改写的有效的内存地址如SHE，RET ADDR，TOP SHE，FUNC HANDLE等周围的内容我们不可控制

- **WINDOWS 2003 堆溢出保护的弱点与问题 (1)**

- ü 保护的范围

跟踪所有的利用途径，发现：

WINDOWS 2003只对出链表的路径做了完善检查，缺乏对入链表路径利用的检查。

构造满足入链表攻击的条件就能成功修改一个特定内存的值为释放堆块的地址。

- ü MS遗漏检查的原因？

- **WINDOWS 2003 堆溢出保护的弱点与问题（2）**

- ü 检查的逻辑

- 逻辑上，并不能真正保证 检查的正确性。

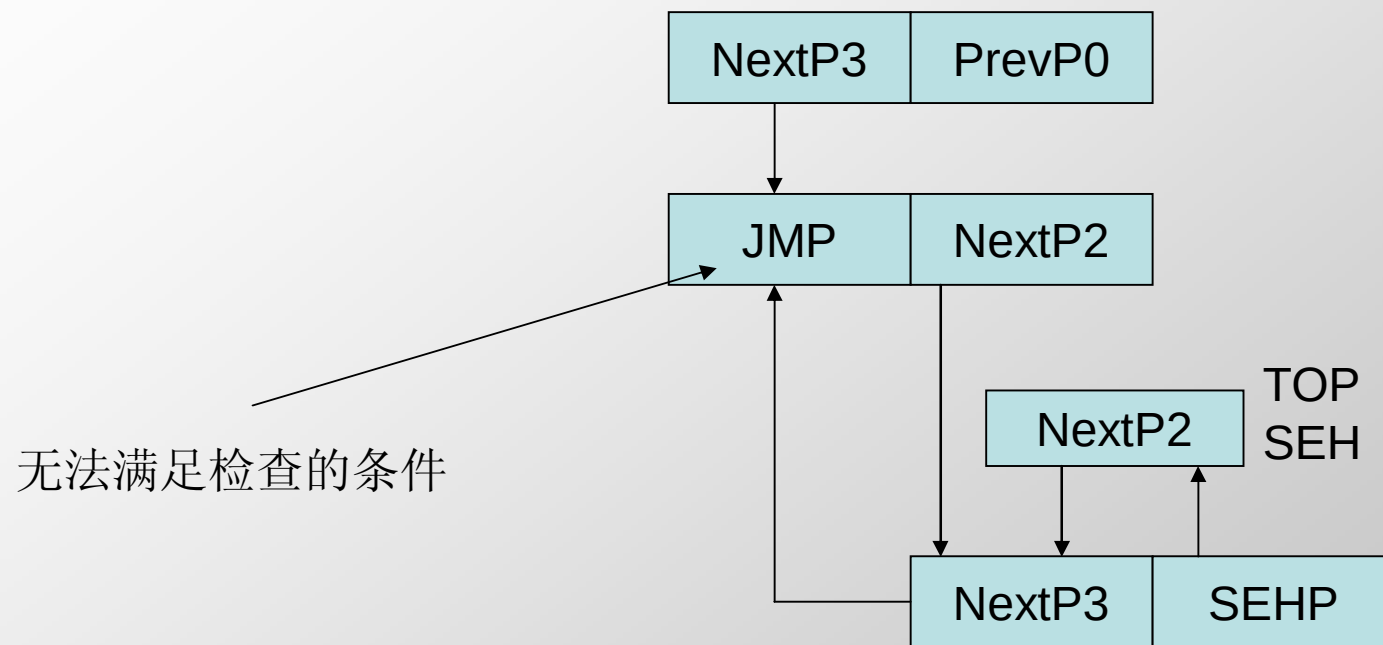
某种含义上，我们还是能欺骗这种保护，如果我们能找到一种有普遍意义的特殊的利用形式，我们也能绕过检查。

- 思路的延续：遗漏检查利用深入的讨论（1）

- ü 新的困难

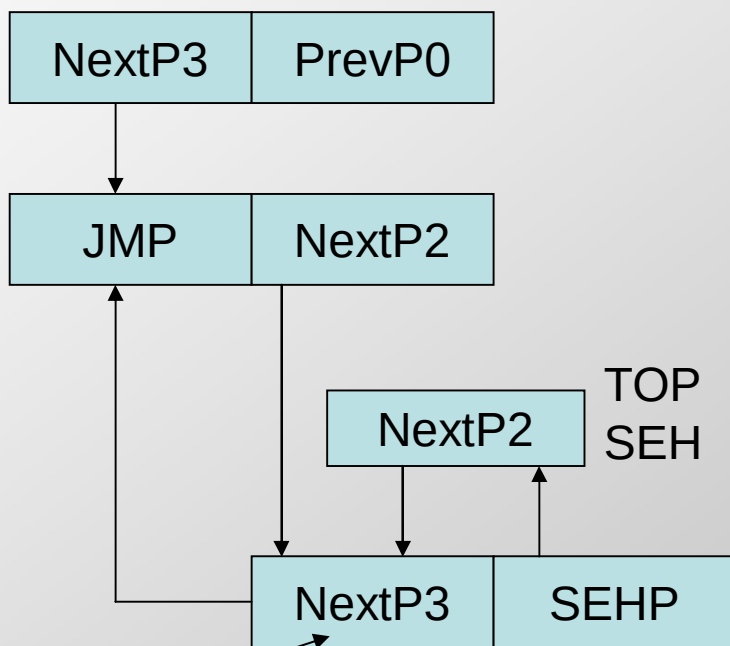
- 无法使用分配的过程再次写入JMP CODE，这样头8字节我们无法控制其内容，引起SHELLCODE执行的异常或失败。

因为分配路径做了完整的保护检查。

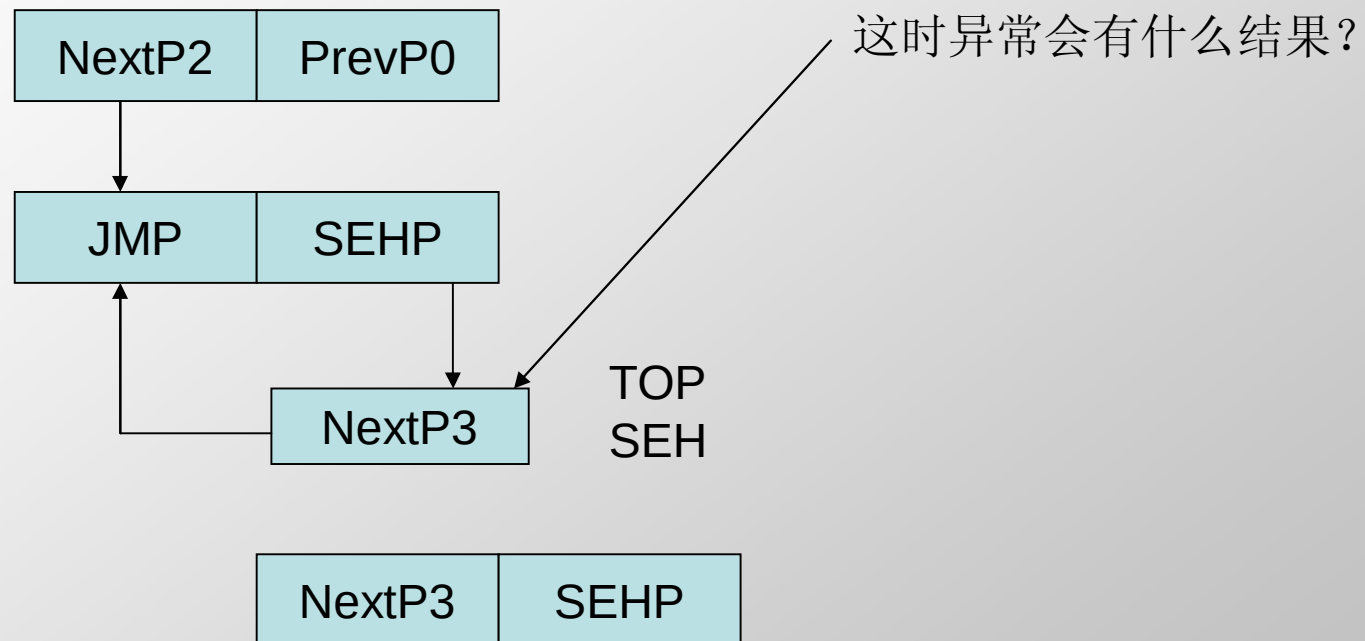


- 思路的延续：遗漏检查利用深入的讨论（2）
 - ü 期待于运气？
 - ü 再次的推导：构造的欺骗
 - 如果我们能知道堆块的准确地址，是否可以构造欺骗？
 - 为什么这里能构造欺骗而前面不能？
 - ü 演示：一个构造欺骗写入JMP的实现

- 思路的延续：遗漏检查利用深入的讨论（3）
 - ü 新的发现与困境
 - 一个可以满足检查的堆块的脱链操作将再次改写SEH



如果我们能让这个堆块进行脱链表操作又如何？



- 思路的延续：遗漏检查利用深入的讨论（4）

- ü 双堆块释放的触发

- 一个可以满足检查的堆块的脱链操作将再次改写**SHE**指向我们原来的被覆盖的堆块（而不是当前释放的堆块），而这里的头4字节是我们可以控制的内容。

- ü 演示：双堆块释放二次重写TOP SHE的实现利用

- 思路的延续：遗漏检查利用深入的讨论（5）

- ü 更深入一步：

- 利用次序，在非双堆块释放环境下构造双堆块释放的环境

- ü 演示：构造双堆块释放的实现

- ü 限制条件汇总

- 思路的突变：更广泛的另类有效利用手段（1）

不能满足于覆盖空闲大堆块的苛刻条件，我们需要再求新的思路

- ü 转换一下思路，构造欺骗用于新的利用形式：

- ü 思路的来源

- 思路的突变：更广泛的另类有效利用手段（2）

- ü 一种利用的转换
与堆栈/数据区的结合
甚至可以用于制造空闲大堆块覆盖的条件

- 思路的突变：更广泛的另类有效利用手段（3）

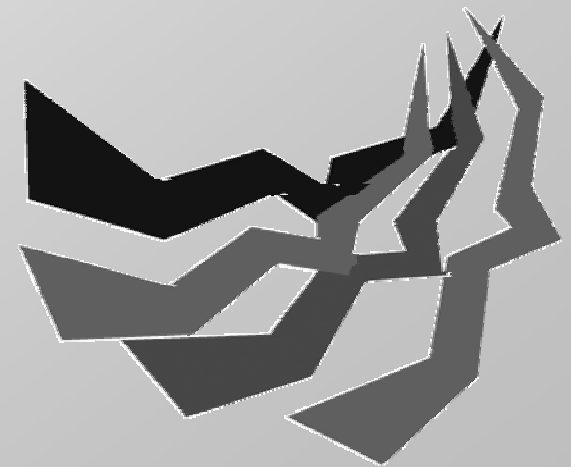
- ü 演示：一个将堆溢出转化为数据区溢出利用的实现

- ü 限制条件汇总

- 未来WINDOWS 2003堆溢出研究的方向

Q/A

Thanks !



X'CON 2003