请谨慎编码，哪怕它只是一句错误处理

— 来自 win32k ! EPATHOBJ::pprFlattenRec 漏洞的启示

wang yu

Hacks in Taiwan, 2013

第一部分

议题简介

议题简介

·关于作者（ wangyu@360.cn ）

·议题背景

2013 年 3 月 Tavis Ormandy 前辈披露了一个微软win32k 模块的问题 —— 在内存压力测试的情况下，例程 win32k!EPATHOBJ::bFlatten 发生了蓝屏。

接下来，关于该蓝屏的利用可谓是一波三折。5 月，随着对链表关系的深度分析，蓝屏问题终于上升为本地提权问题，exploit-db 对此问题的关注度激增。

本议题将聚焦于 win32k!EPATHOBJ 子系统数据结构的设计与实现，以白盒的视角审视上述提权漏洞的原理及利用细节。

议题简介

·议题涵盖

- win32k!EPATHOBJ 子系统相关数据结构的背景、设计与实现
- win32k!EPATHOBJ::pprFlattenRec 漏洞 (CVE-2013-3660) 的产生原因及几种可行的修复方案
- CVE-2013-3660 任意地址写入漏洞的利用思路及演进过程
- 漏洞给我们带来的思考与启示
- 更多测试与审计

·责任声明

第二部分

PATH 子系统设计背景与数据结构

让我们从这里开始

..... EPATHOBJ::pprFlattenRec() is an internal routine for applying this process to
a linked list of PATHRECORD objects.
If you follow the logic, you can see that the PATHRECORD object retrieved from
newpathrec() is mostly initialized, but with one obvious error:

```
EPATHOBJ::pprFlattenRec(_PATHRECORD *)+33:
.text:BFA122CD        mov     eax, [esi+PATHRECORD.prev]              ; load old prev
.text:BFA122D0        push    edi
.text:BFA122D1        mov     edi, [ebp+new]                          ; get the new PATHRECORD
.text:BFA122D4        mov     [edi+PATHRECORD.prev], eax    (1)       ; copy prev pointer over
.text:BFA122D7        lea     eax, [edi+PATHRECORD.count]             ; save address of count member
.text:BFA122DA        xor     edx, edx                                ; set count to zero
.text:BFA122DC        mov     [eax], edx                    (2)
.text:BFA122DE        mov     ecx, [esi+PATHRECORD.flags]             ; load flags
.text:BFA122E1        and     ecx, 0FFFFFFEFh                         ; clear bezier flag
.text:BFA122E4        mov     [edi+PATHRECORD.flags], ecx   (3)       ; copy old flags over
```

The next pointer is never initialized! ..... —— Tavis Ormandy

上下文中 next 指针是什么？未初始化意味着什么？

吾将上下而求索 — PATH 子系统

Windows 图形子系统绘制直线或曲线至少需要哪些要素？让我们从面向对象的角度想想：

颜色？宽度？样式？(像素)点的坐标？点的类型？ .....

Windows 图形子系统是怎么做的：

·The *GDI pen objects* manage pens.
·The *device context objects* manage other line and curve drawing attributes.
·Now we need something to manage geometric shapes. This brings us to the *GDI path objects*.

—— Feng Yuan

GDI path objects —— 坐标点与属性集合的管理者

吾将上下而求索 — PATH 子系统

The Win32 API does not provide any functions to create PATH_TYPE objects directly, although you may have always suspected that some object must be created.

GDI objects need to be selected into a Device Context to be used in GDI drawing calls. A path, unlike other GDI objects, does not have an explicit selection function. It's selected implicitly when it's created.

——《Windows Graphics Programming Win32 GDI and DirectDraw》

这一切让我充满好奇

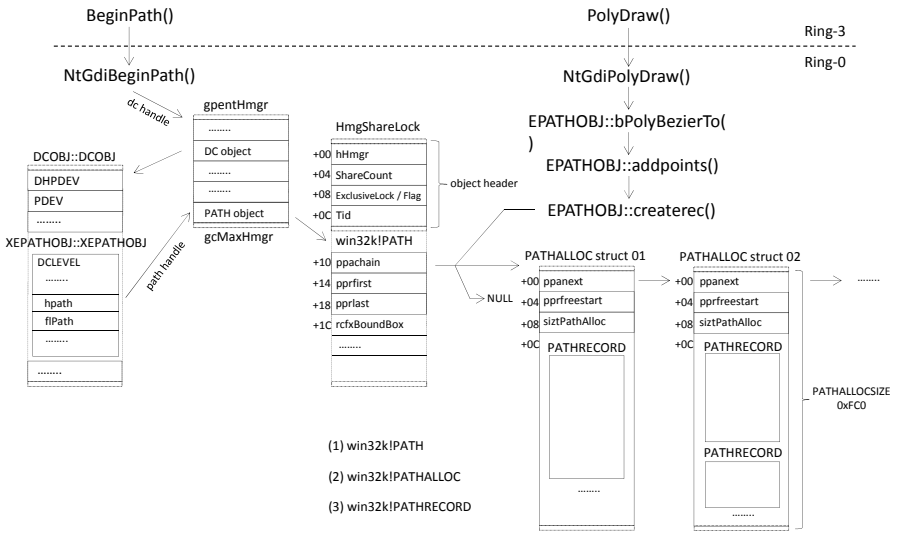
·静态逆向　·动态跟踪　·GDI Debugger Extension

·当然，理论上我还可以... 白盒分析

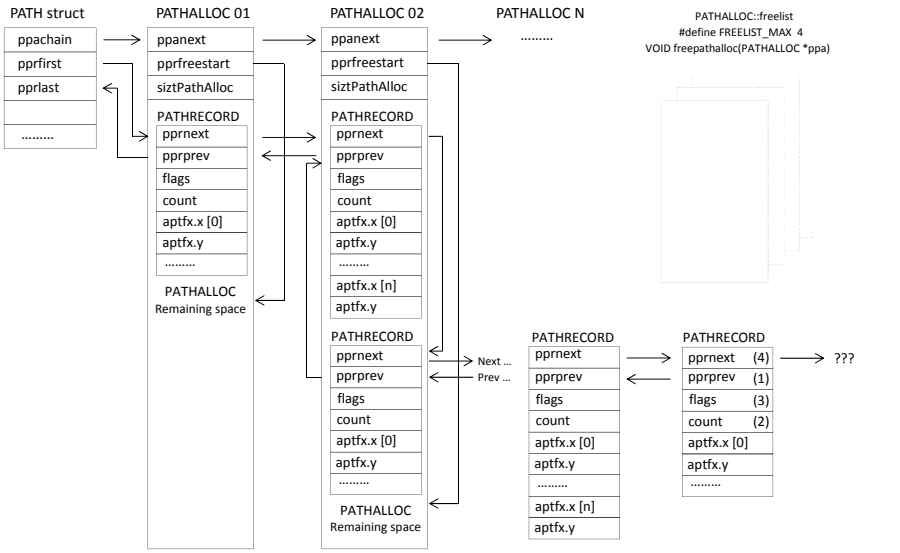
吾将上下而求索 — GDI Debugger Extension

```
Command - Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,reconnect' - WinDbg:6.13.0008.1108 X86
kd> .load D:\Debugging Tools for Windows (x86)\w2kfre\gdikdx.dll
kd> !help
================================================================================
GDIEXTS server debugger extentions:
--------------------------------------------------------------------------------

help
Displays this help page.
All of the debugger extensions support a -? option for extension specifichelp.
All of the debugger extensions that expect a pointer (or handle) can parseexpressions.
such as: ebp+8 or win32k!gpentHmgr
Switches are case insensitive and can be reordered unless otherwise specifiedin the extension help

  - general extensions -

dumphmgr                                -- handle manager objects
dumpdd                                  -- DirectDraw: handle manager objects
dumpobj       [-p pid] [type]           -- all objects of specific type
dumpddobj     [-p pid] [type]           -- DirectDraw: all objects of specific type
dh            [object handle]           -- HMGR entry of handle
dddh          [object handle]           -- DirectDraw: entry of handle
dht           [object handle]           -- handle type/uniqueness/index
dddht         [object handle]           -- DirectDraw: handle type/uniqueness/index
ddc           [DC handle]               -- DC obj (ddc -? for more info)
dpdev         [PDEV ptr]                -- PDEV object (dpdev -? for more info)
dldev         [LDEV ptr]                -- LDEV
dgdev         [GRAPHICS_DEVICE ptr]     -- GRAPHICS_DEVICE
dco           [CLIPOBJ ptr]             -- CLIPOBJ
dpo           [PATHOBJ ptr]             -- PATHOBJ
dppal         [EPALOBJ ptr]             -- EPALOBJ
dpw32         process
```

吾将上下而求索 — PATH 数据结构



吾将上下而求索 — PATH 数据结构



吾将上下而求索 — PATH 数据结构

需要关注的数据结构：

- ·win32k!PATH
- ·win32k!PATHALLOC
- ·win32k!PATHRECORD

简言之，结构 PATHALLOC 是 "PATH 内存池管理器" 的头部，它维护了 "下一链指针"、"剩余空间基址"、"大小" 三项信息。

// 注意，这里的 "大小" 不是指剩余空间的大小，而是池空间的总大小 (PATHALLOCSIZE)

结构 PATHRECORD 是实际的内存使用形式。该池的主要设计目的是存储 _POINTFIX 点信息。在这个漏洞的调用上下文中，"点" 主要被用在贝塞尔曲线的直线化过程 (flattening)。

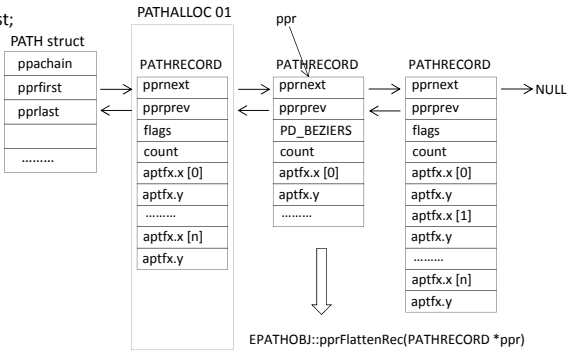
第三部分

漏洞定位

next 指针究竟怎么了？

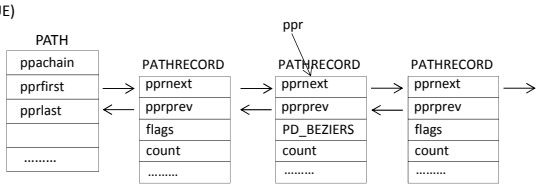
让我们回到例程 EPATHOBJ::pprFlattenRec() 以及它的调用者 EPATHOBJ::bFlatten()，探究 next 指针究竟怎么了。

```
/*
** Cruise over a path, translating all of the  beziers into sequences of lines.
*/

BOOL EPATHOBJ::bFlatten()
{
    for (PATHRECORD *ppr = ppath->pprfirst;
         ppr != (PPATHREC) NULL;
         ppr = ppr->pprnext)
    {
        if (ppr->flags & PD_BEZIERS)
        {
            ppr = pprFlattenRec(ppr);
            ......
        }
    }

    ......

    return TRUE;
}
```



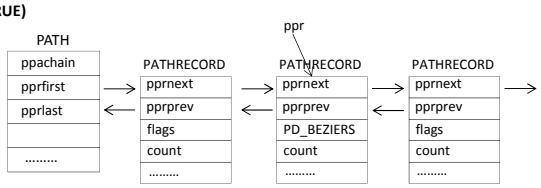
next 指针真的未得到初始化吗？

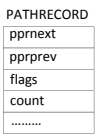
```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```



next 指针真的未得到初始化吗？

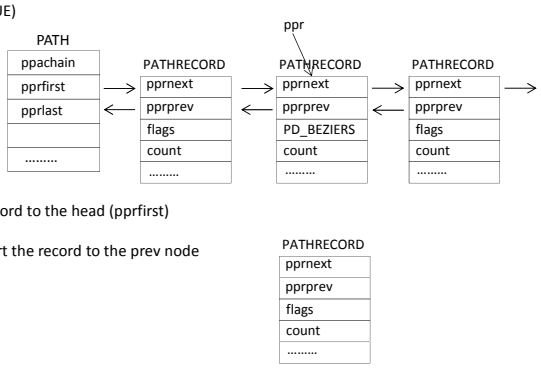
```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```





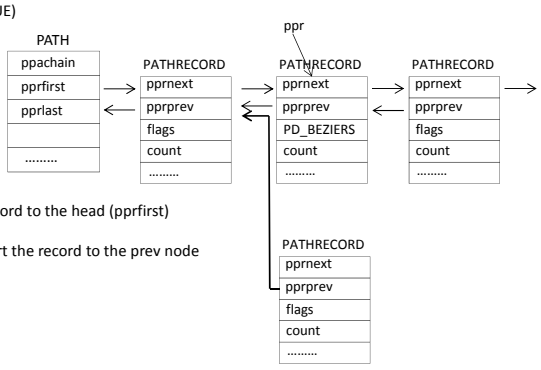
next 指针真的未得到初始化吗？

```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```



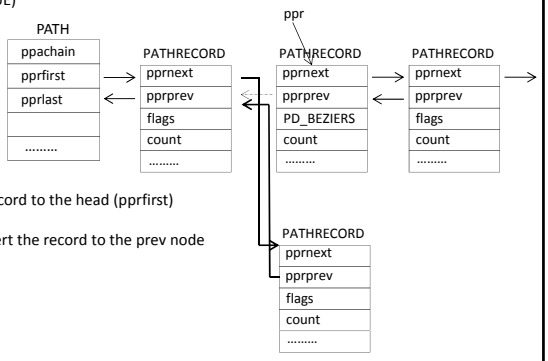
next 指针真的未得到初始化吗？

```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```



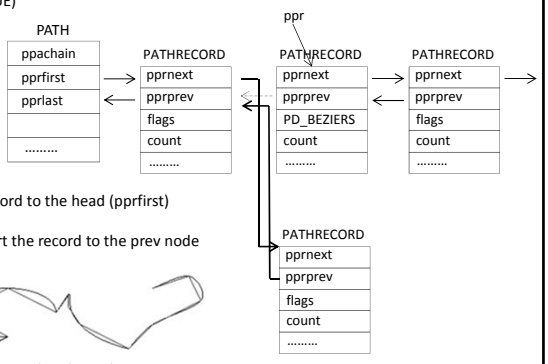
next 指针真的未得到初始化吗？

```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```



next 指针真的未得到初始化吗？

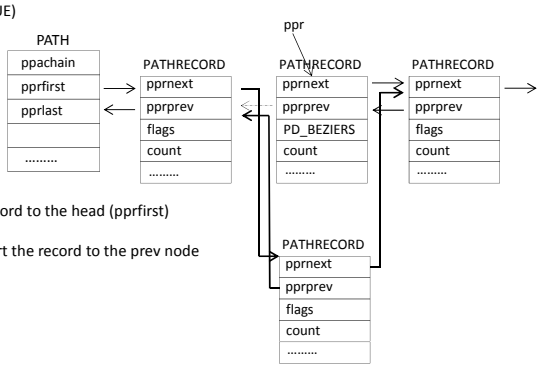
```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```



from GraphicsPath MSDN documentation

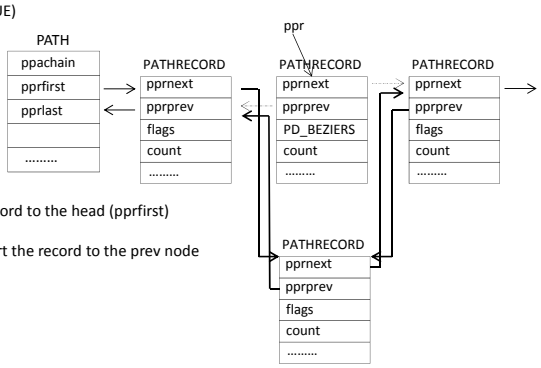
next 指针真的未得到初始化吗？

```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```



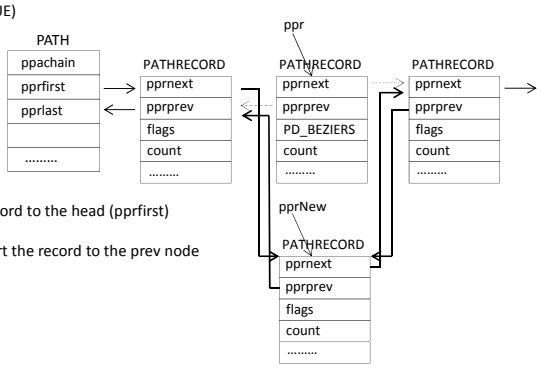
next 指针真的未得到初始化吗？

```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```



next 指针真的未得到初始化吗？

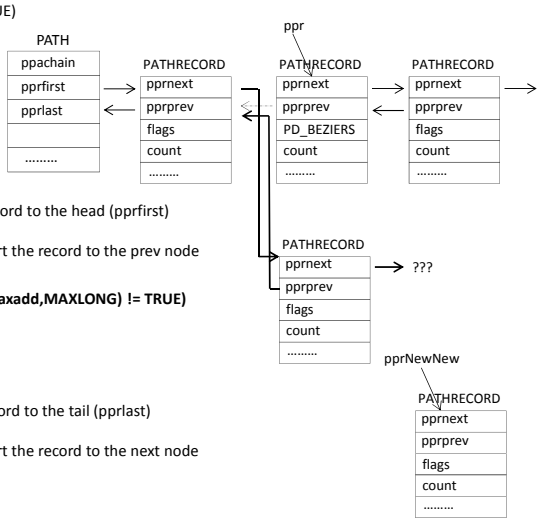
```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```



看起来 pprNew 节点得到了完整的初始化，那么问题究竟藏在哪呢？；)

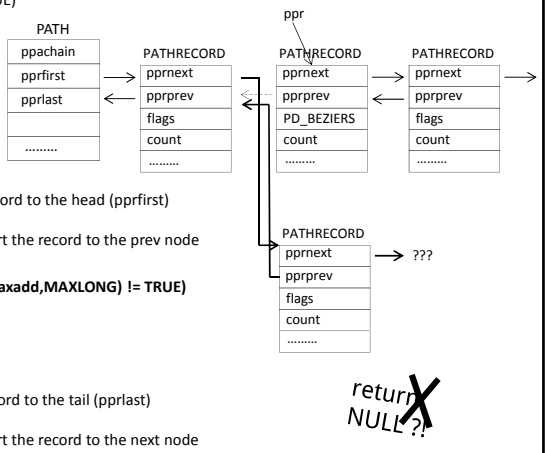
看来我需要仔细审计第四步骤

```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......   if (newpathrec(&pprNewNew,&maxadd,MAXLONG) != TRUE)
                 return (PPATHREC) NULL;

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```



return NULL

```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    // 4. flattening
    ......
    if (newpathrec(&pprNewNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 5. init the next-link of the new record
    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew; // either insert the record to the tail (pprlast)
    else
        pprNew->pprnext->pprprev = pprNew; // or insert the record to the next node

    // 6. return
    return(pprNew);
}
```



问题的根源

win32k!EPATHOBJ::pprFlattenRec 例程的一个分支对于内存分配失败的错误处理代码存在问题。

如果之前的 pprNew 节点是从池中申请的，则该节点的初始内容未知，因为池管理器不会主动内存清零。

如果 pprNewNew 节点内存申请失败，则 PATHRECORD 链表的形态将被破坏 —— 一个初始化了一半的 pprNew 节点被接入了链表，此时未初始化部分 —— pprNew.next 可能是任意值。

遍历上述链表将导致访问 pprNew.next，而谁又知道我们会被引向何方呢？

让我们思考一下，您有几种方案修补这个编码错误？

补丁日前的思考

临时补丁方案 A：主动清零池中的数据

优点：逻辑上容易想到；pprNew.next 如果为 NULL 意味着尾节点
缺点：定位 freepathalloc 等相对复杂；PATHRECORD 链表其它节点丢失

临时补丁方案 B：Patch 池计数器比较代码，禁用池机制

优点：不需要 Inline Hook，1 字节热补丁
缺点：~~PATHALLOC 池机制被禁用；PATHRECORD 链表其它节点丢失~~

正式补丁方案 A：重写错误处理代码

恢复链表正确的形态

正式补丁方案 B：重写链表操作代码

链表操作应保证原子性

微软官方补丁方案

2013 年 6 月 26 日，微软公司发布的 Windows Blue Release Preview Build-9431 版中已经修复了这个问题，这早于 7 月 9 日的补丁日。

Windows 8 9200

```
lea     esi, [ebp+var_F8]
lea     ebx, [ebp+var_100]
mov     edi, edx
mov     [eax+4], ecx
call    ?newpathrec@EPATHOBJ@@IAEHPAPAU_PATHRECORD@@PAKK@Z
cmp     eax, 1
jnz     short loc_1E9801
loc_1E9801:

xor     eax, eax
jmp     loc_117371
loc_117371:                             ; CODE XREF
mov     ecx, [ebp+var_8]
pop     edi
pop     esi
xor     ecx, ebp
pop     ebx
call    @__security_check_cookie@4
leave
retn    4
```

EPATHOBJ::pprFlattenRec(ppr)
CVE-2013-3660
5-Dec-1990 — 9-Jul-2013

Windows Blue RP 9431

```
lea     eax, [ebp-0E8h]
push    eax
lea     eax, [ebp-0ECh]
mov     ecx, edx
push    eax
call    ?newpathrec@EPATHOBJ@@IAEHPAPAU_PATHRECORD@@PAKK@Z
cmp     eax, 1
jnz     short loc_28CCA
loc_28CCA:

xor     ebx, ebx
jmp     loc_8254B
loc_8254B:                              ; CODE XREF
mov     eax, [esi+0Ch]
mov     edx, [ebp+var_E4]
add     eax, 2
lea     ecx, [esi+eax*8]
mov     eax, [edx+8]
mov     eax, [eax+10h]  ; ppath->ppachain->pprfreestart =
mov     [eax+4], ecx                NEXTPATHREC(pprNew);
mov     eax, [edi]
mov     [esi], eax      ; pprNew->pprnext = ppr->pprnext;
test    eax, eax
jz      short loc_825AD ; if (pprNew->pprnext == NULL)
mov     [eax+4], esi    ; pprNew->pprnext->pprprev = pprNew;

loc_8256E:                              ; CODE XREF
mov     eax, ebx
loc_82570:                              ; CODE XREF
mov     ecx, [ebp+var_8]
pop     edi
pop     esi
xor     ecx, ebp
pop     ebx
call    @__security_check_cookie@4
mov     esp, ebp
pop     ebp
retn    4
```

第四部分

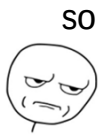
漏洞利用

好吧，这是个问题但那又怎样？

"这没什么大不了吧..."

"这个问题很难触发的..."

"提权漏洞？怎么可能..."



"如果内存申请都失败了，系统早就无法正常工作了..."

可真的是这样吗？

让我们看看一切是如何演进成 "任意地址写入" 的提权问题的。

（1）控制 PATHALLOC 内存池

（2）模拟内存压力测试环境

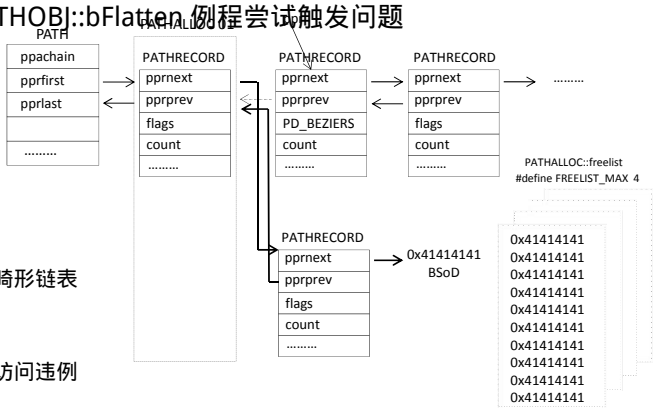
利用第一阶段 — BSoD 演示阶段

http://blog.cmpxchg8b.com/2013/05/introduction-to-windows-kernel-security.html

（1）利用 CreateRoundRectRgn 等技巧大量消耗内存，模拟内存压力测试环境

（2）将垃圾数据以 "点" 的形式压入 PATHALLOC 内存池 —— 池污染

（3）循环调用 EPATHOBJ::bFlatten 例程尝试触发问题



步骤三的一处细节：
3.1 第一次调用
EPATHOBJ::bFlatten：
触发问题代码，构造畸形链表

3.2 第二次调用
EPATHOBJ::bFlatten：
遍历畸形链表，产生访问违例

利用第二阶段 — 扩大可控数据阶段

http://seclists.org/fulldisclosure/2013/May/91

..... The bug is really nice, but exploitation when allocations start failing is tricky. As vuln-dev is dead, I thought I'd post here, I don't have much free time to work on silly Microsoft code, so I'm looking for ideas on how to fix the final obstacle for exploitation .....

将一个用户态地址（而不是垃圾数据）以 "点" 的形式压入 PATHALLOC 池，
win32k!EPATHOBJ::bFlatten() 例程在遍历链表时会访问该可控的环三地址：

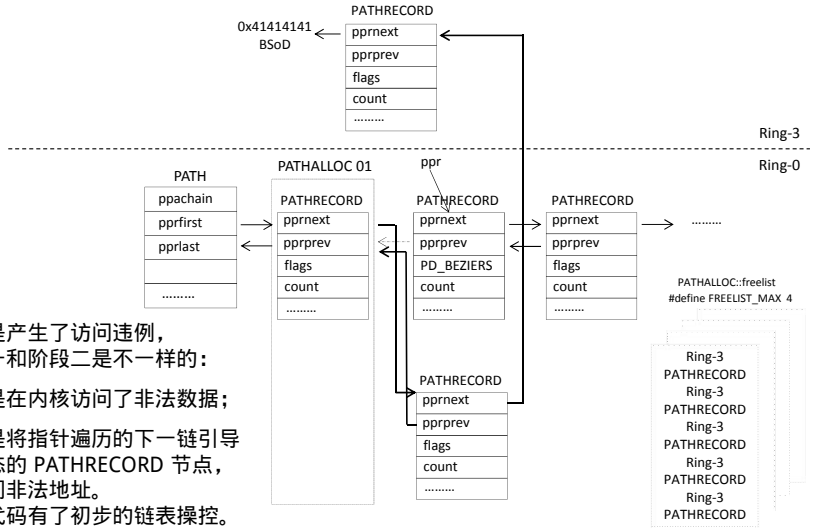
```
// generate a large number of Bezier Curves made up of pointers to our PATHRECORD object.

for (PointNum = 0; PointNum < MAX_POLYPOINTS; PointNum++)
{
    Points[PointNum].x      = (ULONG)(PathRecord) >> 4;
    Points[PointNum].y      = (ULONG)(PathRecord) >> 4;
    PointTypes[PointNum] = PT_BEZIERTO;
}
```

此时的垃圾数据即用户态可控的 PathRecord 节点的基地址，
win32k!EPATHOBJ::bFlatten() 例程访问它意味着将会访问 PathRecord->next

```
PathRecord->next   = (PVOID)(0x41414141);
PathRecord->prev   = (PVOID)(0x42424242);
```

利用第二阶段 — 扩大可控数据阶段

http://seclists.org/fulldisclosure/2013/May/91



虽同样是产生了访问违例，
但阶段一和阶段二是不一样的：

阶段一是在内核访问了非法数据；

阶段二是将指针遍历的下一链引导向用户态的 PATHRECORD 节点，进而访问非法地址。
即攻击代码有了初步的链表操控。

利用第三阶段 — 任意地址写入阶段

http://www.exploit-db.com/exploits/25611/

..... I'm quite proud of this list cycle trick, here's how to turn it into an arbitrary write ...

Tavis Ormandy 为自己自豪的原因是他想到了一个链表循环诡计！想法如下：

仍然将一个环三地址以 "点" 的形式压入 PATHALLOC 池。
环三地址 PathRecord.next 中不再填写无意义的值，而是填写自己：

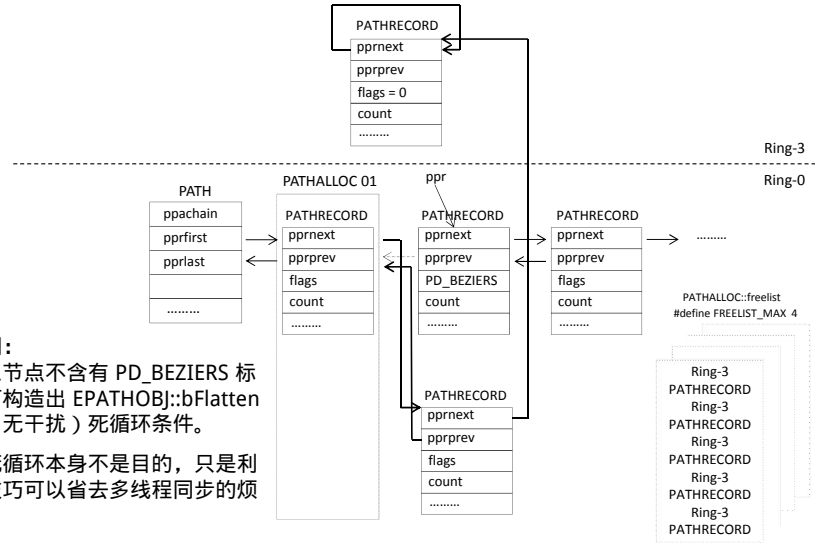
```
PathRecord->next   = PathRecord;
PathRecord->prev   = (PVOID)(0x42424242);
PathRecord->flags  = 0;
```

这个行为会导致 win32k!EPATHOBJ::bFlatten() 遍历链表时产生死循环：

```
BOOL EPATHOBJ::bFlatten()
{
    for (PATHRECORD *ppr = ppath->pprfirst;
         ppr != (PPATHREC) NULL;
         ppr = ppr->pprnext)
    {
        if (ppr->flags & PD_BEZIERS)  // 只要节点不含有 PD_BEZIERS 标志，死循环就立刻成立
        {
            ppr = pprFlattenRec(ppr); // 我们不想此时受到干扰
            .....
        }
    }
    .....
```

利用第三阶段 — 任意地址写入阶段

http://seclists.org/fulldisclosure/2013/May/91

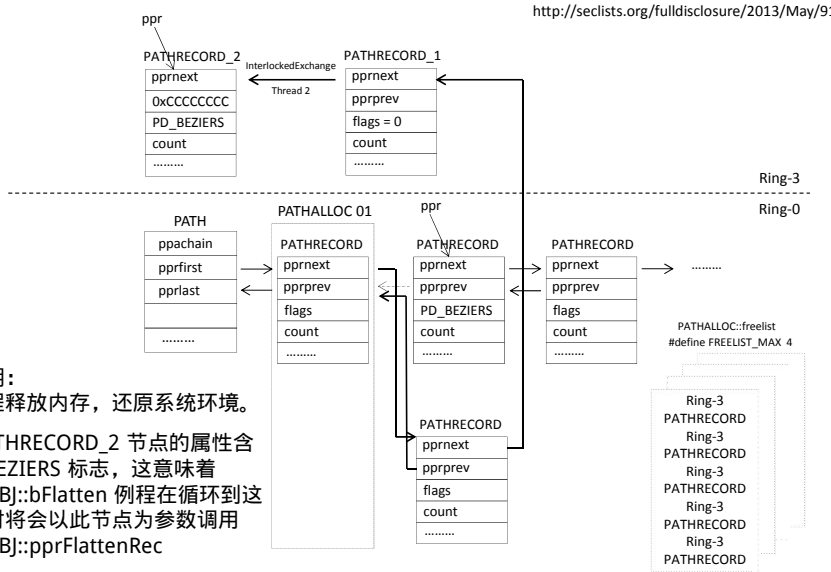


细节说明：
只要环三节点不含有 PD_BEZIERS 标志，就可构造出 EPATHOBJ::bFlatten 例程的（无干扰）死循环条件。

当然，死循环本身不是目的，只是利用这个技巧可以省去多线程同步的烦恼。

利用第三阶段 — 任意地址写入阶段

http://seclists.org/fulldisclosure/2013/May/91

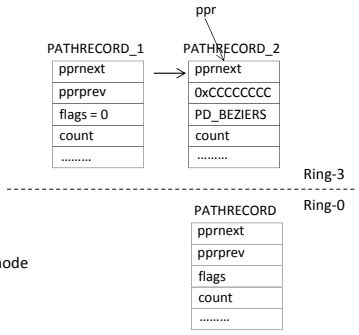


细节说明：
辅助线程释放内存，还原系统环境。

由于 PATHRECORD_2 节点的属性含有 PD_BEZIERS 标志，这意味着 EPATHOBJ::bFlatten 例程在循环到这个节点时将会以此节点为参数调用 EPATHOBJ::pprFlattenRec

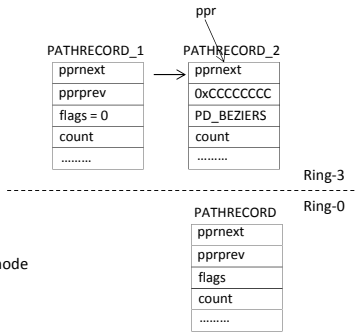
利用第三阶段 — 我们又回来了

```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    ……
}
```



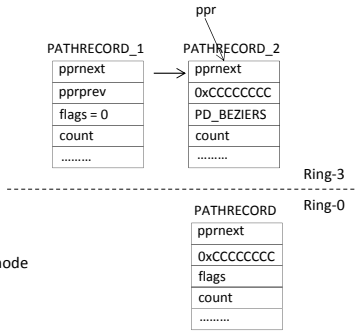
利用第三阶段 — 我们又回来了


```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    ……
}
```



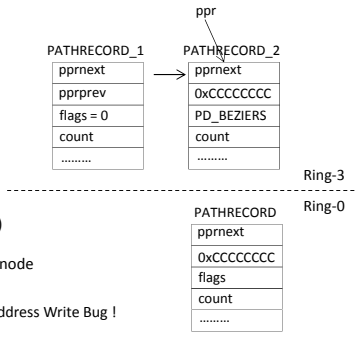
利用第三阶段 — 我们又回来了

```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    ……
}
```



利用第三阶段 — 任意地址写入阶段

```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
    // 1. create a new record
    if (newpathrec(&pprNew,&maxadd,MAXLONG) != TRUE)
        return (PPATHREC) NULL;

    // 2. init some fields (count/flags ...)
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    // 3. init the prev-link of the new record
    pprNew->pprprev = ppr->pprprev;
    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew; // either insert the record to the head (pprfirst)
    else
        pprNew->pprprev->pprnext = pprNew; // or insert the record to the prev node

    ……
}
```



[0xCCCCCCCC + 0] = pprNew; // Arbitrary Address Write Bug !

我们又回到了熟悉的 EPATHOBJ::pprFlattenRec 例程了。还记得我们之前提到过的两次 EPATHOBJ::bFlatten 调用吗？
第一次：触发问题代码，构造畸形链表；第二次：遍历畸形链表，产生访问违例。这里就是利用的完整体现 —— 利用链表赋值操作进行任意地址写入。

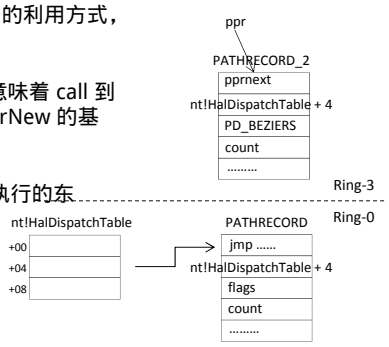
此时利用问题已经转化为传统的 write-4 问题。唯一需要注意的是：任意地址写入后，写入的值不可控，为 pprNew 的基地址。

利用第四阶段 — Home Run！

write-4 问题的利用方案很多，比如参考 ms08-025 的利用方式，将任意地址写的目标指向 nt!HalDispatchTable + 4。

如果有 call [nt!HalDispatchTable+0x04] 的操作即意味着 call 到 pprNew 处，因为（第一次）任意地址写入的是 pprNew 的基址。

所以我们也一定希望 pprNew+0x00 处是一些可以执行的东西，；）
我们有四个字节的发挥空间，
至于是 jmp 0 还是跳转到栈上取参数
pprNew->pprnext = ppr->pprnext; 语句会帮你完成赋值。



唯一需要留神的是 pprNew->pprnext->pprprev = pprNew;
即双链表操作的四个子步骤都需要照顾好，我们要保证第二次 pprNew 写入的页面地址是有写属性的。

x64 的利用方式和上述描述类似。

利用第四阶段 — 攻防永无止尽

jump to nt!HalDispatchTable stack

patch MM_USER_PROBE_ADDRESS /
MM_HIGHEST_USER_ADDRESS
and using (return to) Zero-Page

Mitigation

the others ring-3 shellcode

kernel gadget & ROP

.....

Page Pool NX

NULL-Pointer Dereference

CR4.SMEP

.....

第四部分

思考与启示

思考与启示

漏洞回顾：

程序员编码的疏漏
模拟内存压力环境
→ 有问题的代码被激活
→ 未初始化完成的节点被接入链表
PATHALLOC 内存池的可控特性
→ 执行流程被错误的引导向操作系统用户态
bFlatten 链表遍历代码将未初始化的垃圾数据当做指针来看待

Ring-0
Ring-3

Write-4 任意地址写入
→ 迫使内核函数工作在用户可控参数的环境下
  → 精心构造用户态节点形态
  → 操控内核函数的执行分支
→ 还原内存压力测试环境，使得操作系统重新变得可用

思考与启示

从攻击的角度看待：

坚持不懈，永不言弃，注重积累，将攻击知识融会贯通

从防御的角度看待：

无知者无畏，未知攻焉知防？
千里之堤，毁于蚁穴，任何时候都不能掉以轻心

从阴谋论 / 国家安全的角度看待：

输出的产品有必要内置后门代码吗？太容易被发现了吧
往往一句看似漫不经心的代码就可以打破整个安全模型

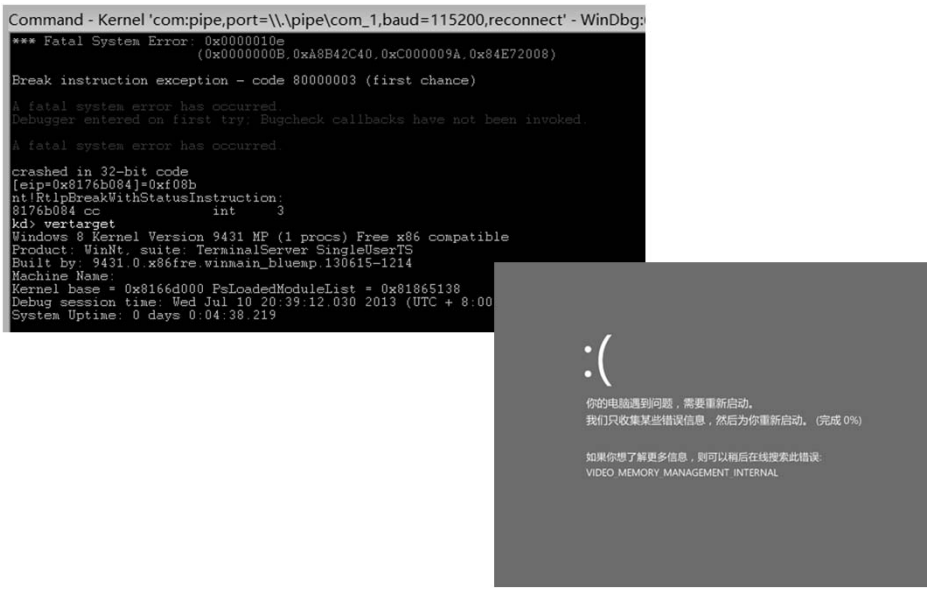
思考与启示

从程序员的角度看待：

请谨慎编码，
哪怕它只是一句错误处理，
哪怕它可以潜藏 8252 天。

EPATHOBJ::pprFlattenRec(ppr)
CVE-2013-3660
5-Dec-1990 — 9-Jul-2013

Dig-it！



致谢！

P1P1Winner PJF Bugvuln RoyceLu

YaoTong PaulFan MJ0011

HIT Committee guys in the 2B# - 14F

Tavis Ormandy

Q&A

wangyu@360.cn