深入理解 Windows 字体解析引擎漏洞

wang yu

SyScan(+)360, 2012

第一部分

议题简介

议题简介

- 关于作者 （ wangyu@360.cn ）
- 议题背景

2011 年 12 月微软月度安全更新修复了此前曾被Duqu 恶意软件利用的 Win32K 内核漏洞。

同月，在中国更受关注的一个话题是网站后台数据库的安全性问题。

本议题将聚焦于 Win32K 字体解析引擎的设计与实现，以白盒的视角审视 Duqu 0-day 的利用细节。

议题简介

- 议题涵盖
  - 字体解析引擎客户端接口（ Font Scaler Client Interface ）的背景、设计与实现
  - 演示如何在系统用户态实现字体引擎（ Font Scaler ）的客户端 —— 引擎的反内核化示例
  - 作为系统内核态字体引擎的客户端，Win32K 模块是如何与之交互的 —— Win32K 的调用假设
  - Duqu 与 MS11-087 远程可执行漏洞
  - 字体引擎的更多审计
- 责任声明

第二部分

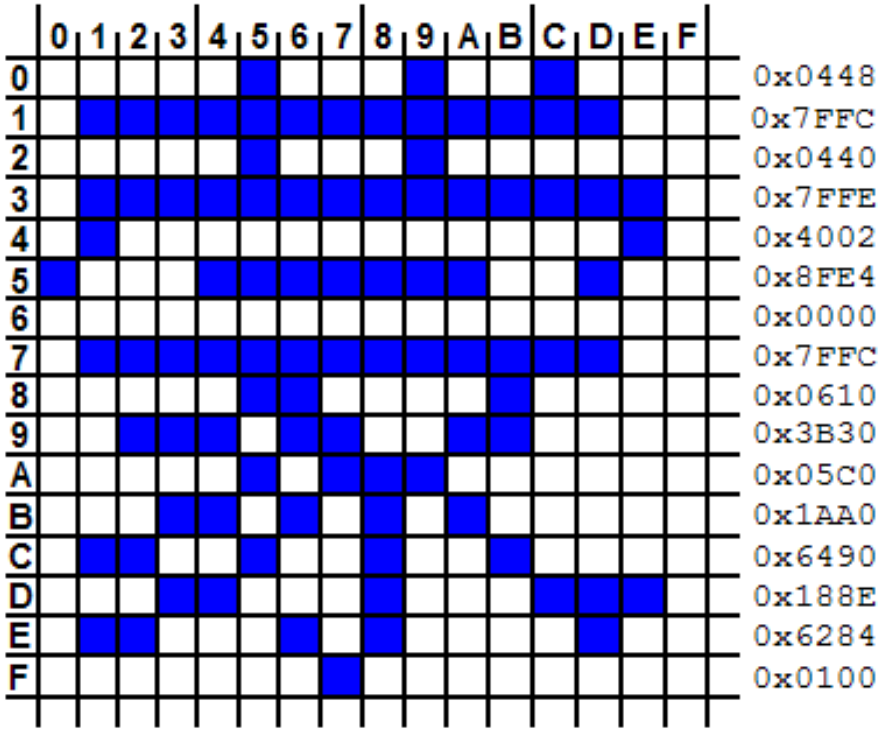
从点阵字体到轮廓字体

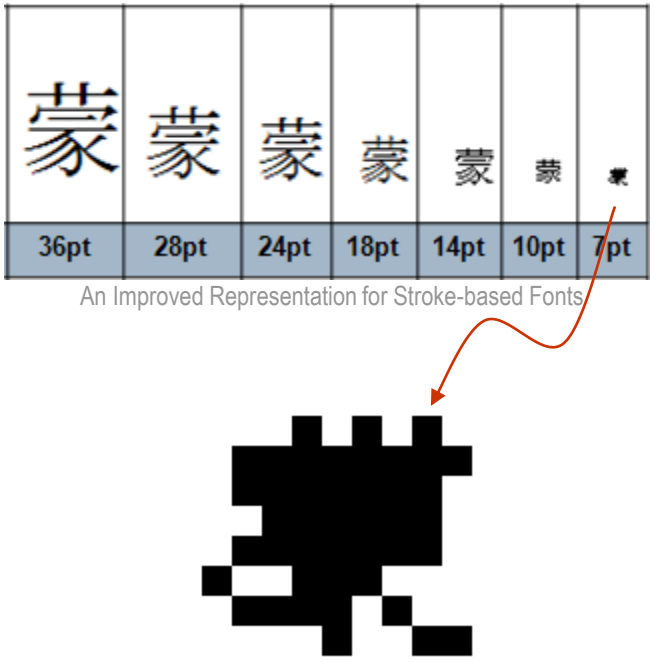
启动扇区里的小游戏

演示：640*480*16 色图形模式写方式二 —— 写点

从点阵字体到轮廓字体

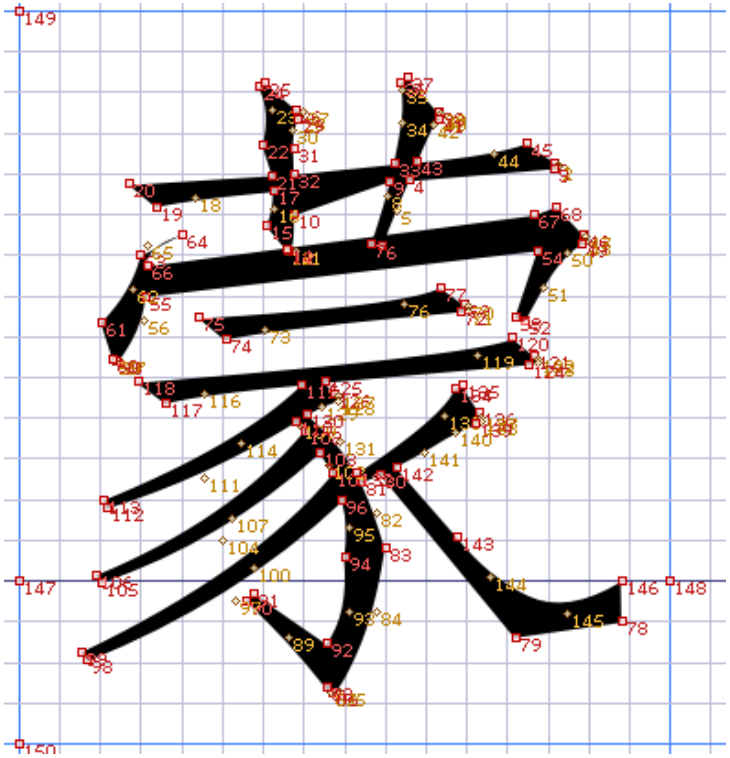
点阵位图 —— 优点与缺点

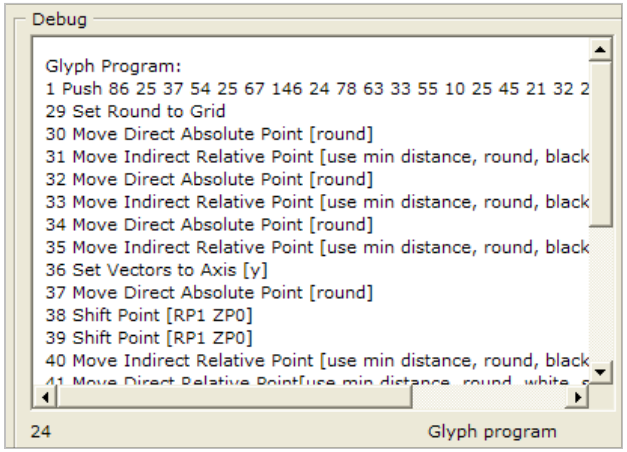


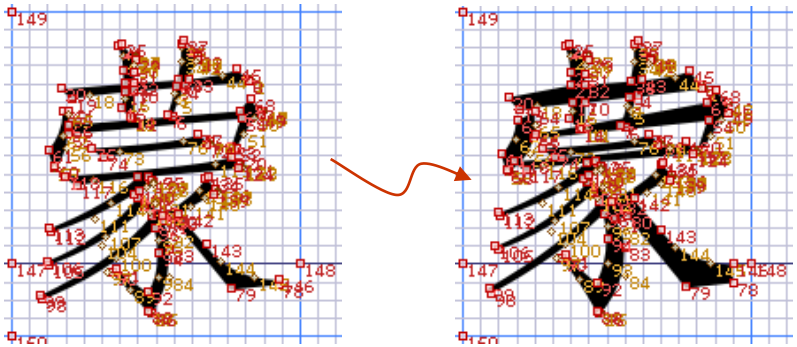


从点阵字体到轮廓字体

轮廓字体 —— 优点与缺点







从点阵字体到轮廓字体

- 我们站在巨人的肩膀之上

  数字字体的混沌时期

  从复印机到 PostScript 页面描述语言，从 Xerox 到 Adobe

  苹果公司的加入 — LaserWriter，1985 年

  从操作系统的角度考虑，苹果公司从上世纪八十年代末开始研发自己的可缩放字体技术 — Royal，这即 TrueType 的前身

- 两种流派，两种理念

  - PostScript Type 1 : cubics; "smarter" fonts, "dumber" interpreter
  - TrueType : quadratics; "dumber"

第三部分

字体解析引擎客户端接口

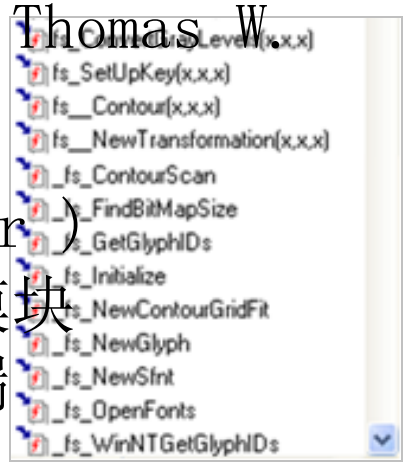
引擎起源

- PostScript Type 1 vs. Royal （ TrueType ）

  PostScript 早于后者六年

- Royal （ TrueType ） vs. TrueImage

  "Apple traded the technology to Microsoft in exchange for the latter 's PostScript clone technology 'TrueImage' ... which was buggy at the time, and never used by Apple"... — Thomas W. Phinney



- 内核化后，TrueType 字体引擎 （ Font Scaler ）
  实现于 Win32K 模块的内部；而 Win32K 模块
  也可被视为字体引擎的调用者或引擎客户端

- 引擎的导出接口即 Font Scaler Client Interface

目标研究手段

- Duqu 0-day 让我充满好奇

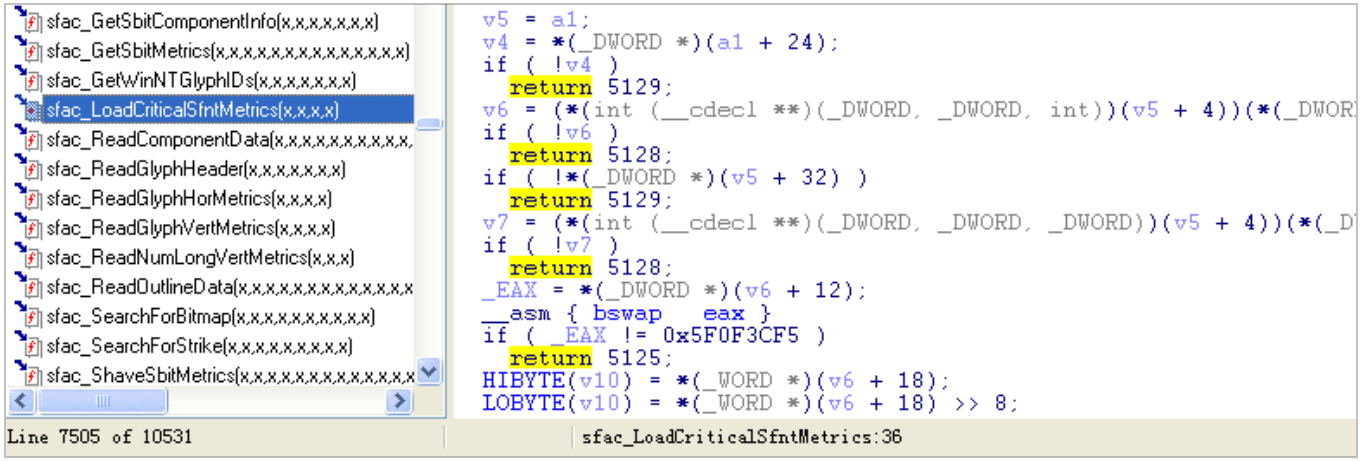
  ″Initially, it even caused confusion among researchers who believed Duqu was exploiting a vulnerability in the MS Word format itself″...

  — Ivan Teblin

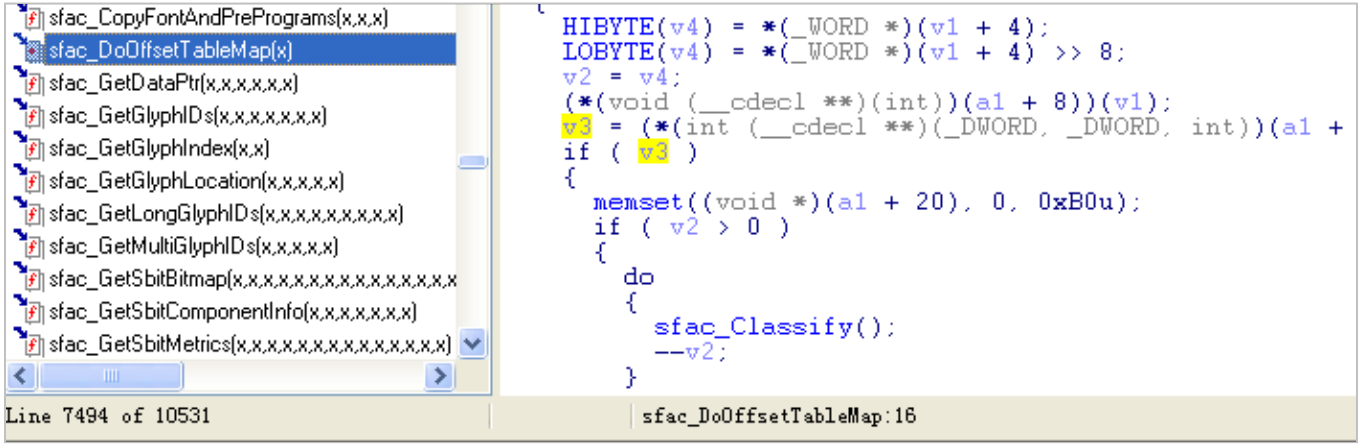
- 静态逆向
- 动态跟踪
- 当然，理论上我还可以... 白盒分析！

还具备参考价值吗？

白盒评估 —从宏观角度（引擎架构）从微观角度（代码笔误）

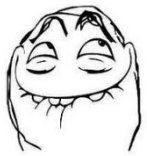


sfac_LoadCriticalSfntMetrics – 6.2.9200.16384 sfntaccs.c line:953



sfac_DoOffsetTableMap sfntaccs.c line:252

还可以工作吗？



I’m Feeling Lucky!

```
c:\project\ntgdi\fondrv\tt\scaler\fscaler.c
/*
** this guy asks for memmory for points, instructions, fdefs and idefs
*/

FS_PUBLIC FS_ENTRY FS_ENTRY_PROTO fs_NewSfnt (fs_GlyphInputType *inputPtr, fs_GlyphInfoType *outputPtr)
{
    ErrorCode       error;
    fs_SplineKey*   key;

        CHECKSTAMP(inputPtr->memoryBases[KEY_PTR_BASE] + outputPtr->memorySizes[KEY_PTR_BASE]);

    STAT_ON_NEWSFNT; /* start STAT timer */

    key = fs_SetUpKey(inputPtr, INITIALIZED, &error);

    if (!key)
```

```
Command
eax=0006ff48 ebx=00000000 ecx=04018080 edx=00000000 esi=000829a0 edi=7c80ac61
eip=040082ce esp=0006ff40 ebp=0006ff4c iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
fuzz!fs_NewSfnt+0xe:
040082ce 33d2            xor     edx,edx
0:000> p
eax=0006ff48 ebx=00000000 ecx=04018080 edx=00000000 esi=000829a0 edi=7c80ac61
eip=040082d0 esp=0006ff40 ebp=0006ff4c iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
fuzz!fs_NewSfnt+0x10:
040082d0 e8abfeffff      call    fuzz!fs_SetUpKey (04008180)
```

工欲善其事 必先利其器

```
C:\>ttfdump.exe

; TrueType v1.0 Dump Program - v1.8, Oct 29 2002, rrt, dra, gch, ddb, lcp, pml
; Copyright (C) 1991 ZSoft Corporation. All rights reserved.
; Portions Copyright (C) 1991-2001 Microsoft Corporation. All rights reserved.

Usage: TTFDUMP <filename> [-nNNNN] [-tCCCC] [-h] [-zHHH] [-cNNNN] [-q]
        <filename>  - TrueType .TTF (or .T2 or .ROY or .TTC) filename
```

字体格式分析工具

TTFDump (FontTools)

http://www.microsoft.com/typography/tools/tools.aspx

```
Name ID:          7
Length:           98
Offset:           180
   Data:  0 44  0 65  0 78  0 74  0 65  >   -D.e.x.t.e
          0 72  0 20  0 69  0 73  0 20  >   -r. .i.s.
          0 61  0 20  0 72  0 65  0 67  >   -a. .r.e.g
          0 69  0 73  0 74  0 65  0 72  >   -i.s.t.e.r
          0 65  0 64  0 20  0 74  0 72  >   -e.d. .t.r
          0 61  0 64  0 65  0 6D  0 61  >   -a.d.e.m.a
          0 72  0 6B  0 20  0 6F  0 66  >   -r.k. .o.f
          0 20  0 53  0 68  0 6F  0 77  >   - .S.h.o.w
          0 74  0 69  0 6D  0 65  0 20  >   -t.i.m.e.
          0 49  0 6E  0 63  0 2E        >   -I.n.c..
```

```
'OS/2' Table - OS/2 and Windows Metrics
---------------------------------------
Size = 86 bytes (expecting 86 bytes)
  'OS/2' version:              1
  xAvgCharWidth:               2048
  usWeightClass:               400
  usWidthClass:                5
  fsType:                      0x0008
 ySubscriptXSize:              1
 ySubscriptYSize:              1
 ySubscriptXOffset:            0
 ySubscriptYOffset:            0
 ySuperscriptXSize:            1
 ySuperscriptYSize:            1
 ySuperscriptXOffset:          0
 ySuperscriptYOffset:          0
 yStrikeoutSize:               1
 yStrikeoutPosition:           0
 sFamilyClass:                 0      subclass = 0
 PANOSE:                       0  0  0  0  0  0  0  0  0  0
 Unicode Range 1( Bits 0 - 31 ): 00000001
 Unicode Range 2( Bits 32- 63 ): 00000000
 Unicode Range 3( Bits 64- 95 ): 00000000
 Unicode Range 4( Bits 96-127 ): 00000000
```

工欲善其事 必先利其器

呃... TTFDump 唯一的问题就是问题太多

```
eax=00000000 ebx=deadbeef ecx=2b2e2c31 edx=00381bb9 esi=00380f68 edi=00000000
eip=00401f9d esp=0012fe34 ebp=00380f60 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
ttfdump+0x1f9d:
00401f9d 8a242b          mov     ah,byte ptr [ebx+ebp]      ds:0023:dee5ce4f=??
```

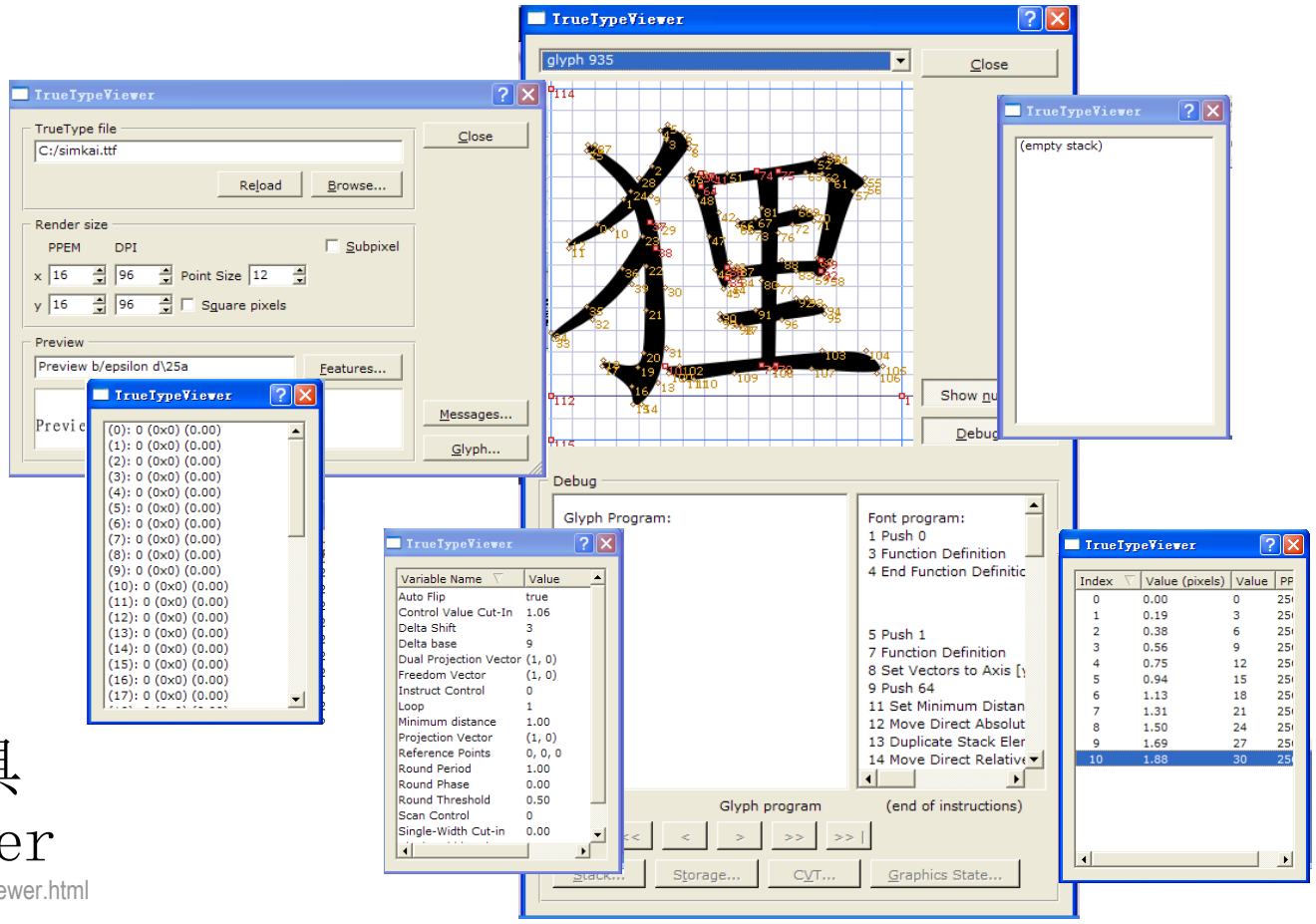
```
eax=0000001c ebx=00030003 ecx=475df354 edx=00381feb esi=00000001 edi=0046004b
eip=004043e6 esp=0012fe08 ebp=00430170 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
ttfdump+0x43e6:
004043e6 660fb64f03      movzx   cx,byte ptr [edi+3]        ds:0023:0046004e=??
```

```
eax=00000000 ebx=00430048 ecx=00000000 edx=28000000 esi=f3740001 edi=27ffffff
eip=00404447 esp=0012fdf8 ebp=004300b0 iopl=0         nv up ei pl nz na pe cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000207
ttfdump+0x4447:
00404447 8b441f04        mov     eax,dword ptr [edi+ebx+4] ds:0023:2843004b=????????
```

```
eax=00000042 ebx=0000ffff ecx=5f0b4a14 edx=00381b4c esi=00384000 edi=00000c1f
eip=00403774 esp=0012fddc ebp=00000021 iopl=0         nv up ei ng nz na po cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000283
ttfdump+0x3774:
00403774 0fb64603        movzx   eax,byte ptr [esi+3]       ds:0023:00384003=??
```

```
eax=00382c78 ebx=00380000 ecx=00003801 edx=78003801 esi=00382c70 edi=00382c30
eip=7c930a19 esp=0012fd44 ebp=0012fd50 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
ntdll!RtlpCoalesceFreeBlocks+0x36e:
7c930a19 8b09            mov     ecx,dword ptr [ecx]  ds:0023:00003801=????????
0:000> kb
ChildEBP RetAddr  Args to Child
0012fd50 7c93084c 00380000 00003801 0012fe08 ntdll!RtlpCoalesceFreeBlocks+0x36e
0012fe24 0040f217 00380000 00000000 00382c38 ntdll!RtlFreeHeap+0x2e9
0012fe38 00409097 00382c38 00382ae0 00382c30 ttfdump+0xf217
0012fe64 0040a1d9 00000003 00382ae0 02000006 ttfdump+0x9097
0012fee4 0040fb89 00000000 00380f90 00380fe8 ttfdump+0xa1d9
00000000 00000000 00000000 00000000 00000000 ttfdump+0xfb89
```

工欲善其事 必先利其器



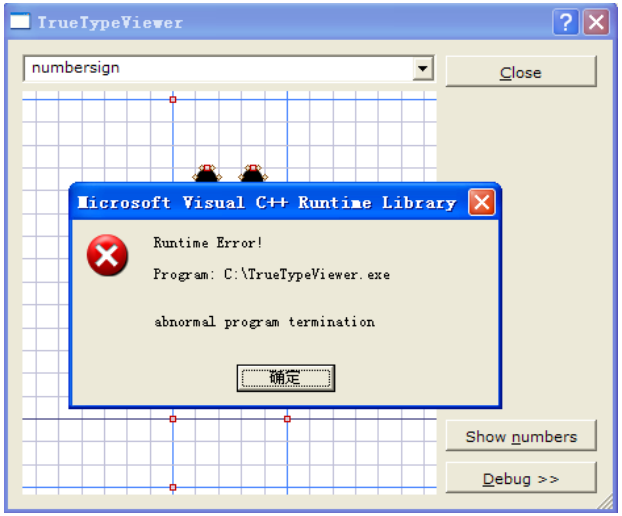
字体调试工具

TrueTypeViewer

http://home.kabelfoon.nl/~slam/fonts/truetypeviewer.html

工欲善其事 必先利其器

囧... 好的



引擎的重要接口

Font Scaler Client Interfa

例程名称	功能描述
fs_OpenFonts	为 CJ_OUT.KEY_PTR_BASE 预定空间， 预定大小为 sizeof(fs_SplineKey) + 校验标志域， 对解析引擎而言该例程是必须的
fs_Initialize	初始化 CJ_0（fs_SplineKey 结构）的部分域， 如 TransformInfoSubPixel，初始化 BitMask 等 对解析引擎而言该例程是必须的
fs_NewSfnt	初始化 fs_SplineKey.ClientInfo 的关键外部回调， 读取并填充 TableDirectory、MaxProfile 等域 为 CJ_3、CJ_4 预定空间，读取 cmap 表等
fs_NewTransformation	初始化 CJ_3（WORK_SPACE_BASE） 初始化 CJ_4（PRIVATE_FONT_SPACE_BASE） 构建指令执行环境，执行 Pre、FontProgram
fs_NewGlyph	根据 charCode 定位 Index（或由调用者指定） 判断目标字形是否存在内嵌位图数据
fs_ContourNoGridFit	按指定磅值构建字形轮廓，轮廓不需要网格适配
fs_ContourGridFit	按指定磅值构建字形轮廓，轮廓需要网格适配 图元修正指令

表 Font Scaler Client Interface

引擎的重要接口

fs_FindBitMapSize	计算待显示字形的位图总字节数
fs_SaveOutlines	将轮廓数据保存至轮廓数据缓存 对解析引擎而言该例程是可选的
fs_RestoreOutlines	从轮廓数据缓存中恢复轮廓数据 对解析引擎而言该例程是可选的
fs_ContourScan	将轮廓信息转换为位图 (光栅化)
fs_CloseFonts	关闭字体解析引擎

续表 Font Scaler Client Interface

例程前缀功能描述

例程前缀	功能描述
fs_	引擎导出接口
fs__	引擎内部封装
fsc_	引擎转换 (Converter) 例程
fsg_	引擎支持例程
sbit_	位图处理例程
itrp_	指令虚拟机支持例程
sfac_	字体结构解析例程
mth_	数学运算例程
fnterr_	错误处理例程

表 Routine Prefix

引擎的核心数据结构

引擎输入、输出核心结构

fs_GlyphInputType	CJ_IN
fs_GlyphInfoType	CJ_OUT
fs_SplineKey	CJ_0
fsg_WorkSpaceOffsets	CJ_3
fsg_PrivateSpaceOffsets	CJ_4

图形状态核心结构

fnt_LocalGraphicStateType

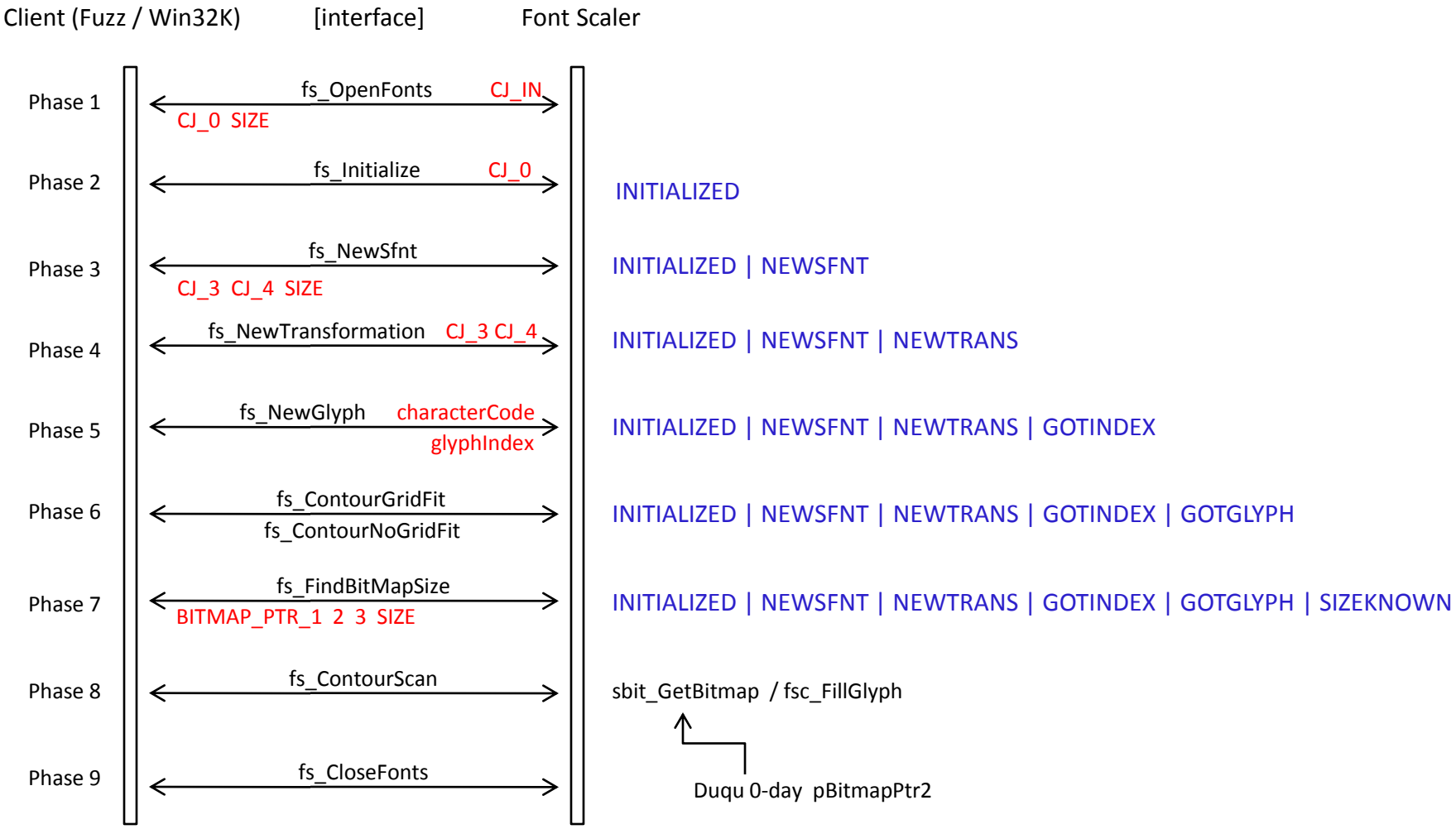
fnt_ElementType

fnt_GlobalGraphicStateType

因为白盒，所以不在话下

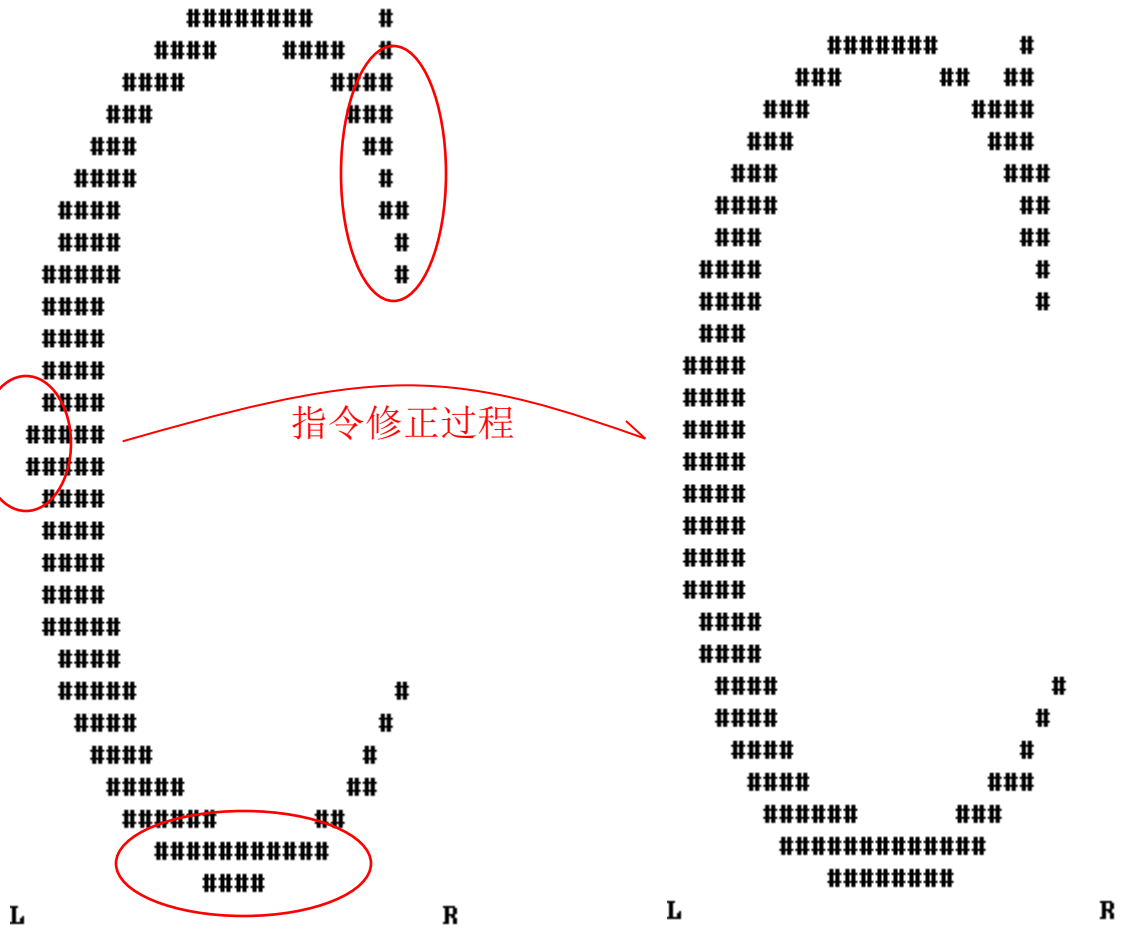
```
0:000> dt fnt_GlobalGraphicStateType
fuzz!fnt_GlobalGraphicStateType
   +0x000 stackBase          : Ptr32 Int4B
   +0x004 store              : Ptr32 Int4B
   +0x008 controlValueTable  : Ptr32 Int4B
   +0x00c pixelsPerEm        : Uint2B
   +0x00e pointSize          : Uint2B
   +0x010 fpem               : Int4B
   +0x014 engine             : [4] Int4B
   +0x024 defaultParBlock    : fnt_ParameterBlock
   +0x058 localParBlock      : fnt_ParameterBlock
   +0x08c funcDef            : Ptr32 fnt_funcDef
   +0x090 instrDef           : Ptr32 fnt_instrDef
   +0x094 ScaleFuncXBase     : Ptr32     long
   +0x098 ScaleFuncYBase     : Ptr32     long
   +0x09c ScaleFuncX         : Ptr32     long
   +0x0a0 ScaleFuncY         : Ptr32     long
   +0x0a4 ScaleFuncCVT       : Ptr32     long
   +0x0a8 pgmList            : [2] fnt_pgmList
   +0x0b8 scaleXBase         : fnt_ScaleRecord
   +0x0c8 scaleYBase         : fnt_ScaleRecord
   +0x0d8 scaleX             : fnt_ScaleRecord
   +0x0e8 scaleY             : fnt_ScaleRecord
   +0x0f8 scaleCVT           : fnt_ScaleRecord
   +0x108 cvtStretchX        : Int4B
   +0x10c cvtStretchY        : Int4B
   +0x110 identityTransformation : Char
   +0x111 non90DegreeTransformation : Char
   +0x114 xStretch           : Int4B
   +0x118 yStretch           : Int4B
   +0x11c init               : Char
   +0x11d pgmIndex           : UChar
   +0x120 instrDefCount      : Int4B
   +0x124 bSameStretch       : UChar
   +0x125 bCompositeGlyph    : UChar
   +0x128 maxp               : Ptr32 LocalMaxProfile
   +0x12c cvtCount           : Uint2B
   +0x130 interpScalarX      : Int4B
   +0x134 interpScalarY      : Int4B
   +0x138 fxMetricScalarX    : Int4B
   +0x13c fxMetricScalarY    : Int4B
```

引擎的基本执行流



启动引擎！

演示：字体引擎的反内核化示例



引擎的设计准则与背景推测

- 引擎最初的工作模式
- 引擎对于内存的使用策略
- “高内聚/低耦合”或是“低内聚/高耦合”？
- 毫无疑问，这是一个时代的产物

第四部分

Win32K 的假设与 MS11-087

现有议题资料

- CVE-2011-3402 分析, venustech
- GDI Font Fuzzing in Windows Kernel for Fun, bh12, PacSec12
- Anatomy of Duqu exploit, vb100
- The Cousins of Stuxnet: Duqu, Flame, and Gauss, CrySyS Lab

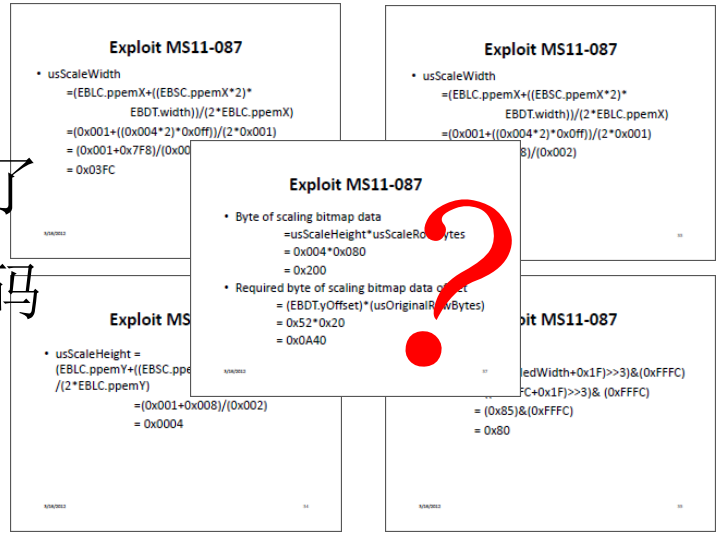
- 共识：

  某处发生了越界导致某个值被改写了

  精心构造的字体指令触发了恶意代码

- 疑问：

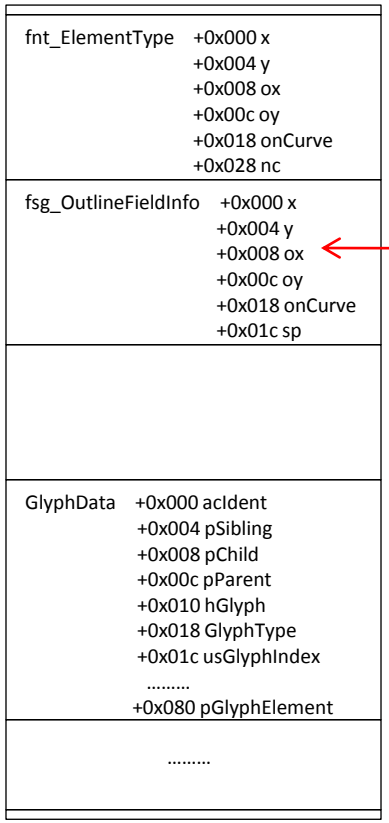
  太多的细节值得深入挖掘



CJ_3 和 CJ_4

typedef struct fsg_WorkSpaceOffsets aka. CJ_3 WORKSPACE MEMORY LAYOUT

{



[WorkSpaceOffsets->ulGlyphElementOffset]

TrueType Glyph Element [MaxComponentDepth + 1]

sizeof(fnt_ElementType) * (MAX_COMPONENT_DEPTH(pMaxProfile) + 1)

[WorkSpaceOffsets->ulGlyphOutlineOffset]

TrueType Glyph Outline

WorkSpaceOffsets->ulReusableMemoryOffset

(WorkSpaceOffsets->ulReusableMemoryOffset += WorkSpaceOffsets->ulGlyphOutlineOffset)

[WorkSpaceOffsets->ulGlyphDataByteSetOffset]

TrueType Glyph Data Allocation ByteSet

MAX_NESTED_GLYPHS(MaxProfile) * sizeof(boolean)

[WorkSpaceOffsets->ulGlyphDataOffset]

TrueType GlyphData [ulGlyphDataCount]

plExtraWorkSpace

MAX_NESTED_GLYPHS(MaxProfile) * sizeof(GlyphData)

[WorkSpaceOffsets->ulStackOffset]

TrueType Stack

MaxProfile->maxStackElements * sizeof(F26Dot6)

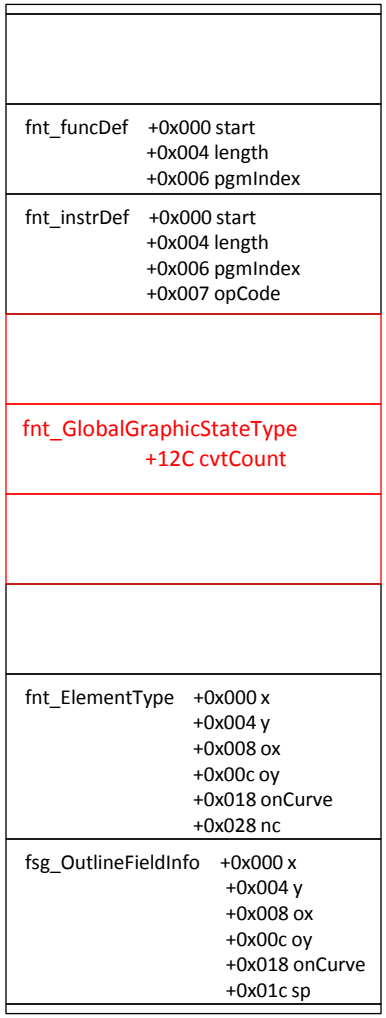
[END]

}

CJ_3 和 CJ_4

typedef struct fsg_PrivateSpaceOffsets aka. CJ_4 PRIVATE SPACE MEMORY LAYOUT

{



[PrivateSpaceOffsets->offset_storage]

TrueType Storage

sizeof(F26Dot6) * MaxProfile->maxStorage

[PrivateSpaceOffsets->offset_functions]

TrueType Function Defs

sizeof(fnt_funcDef) * MaxProfile->maxFunctionDefs

[PrivateSpaceOffsets->offset_instrDefs]

TrueType Instruction Defs

sizeof(fnt_instrDef) * MaxProfile->maxInstructionDefs

[PrivateSpaceOffsets->offset_controlValues]

TrueType Scaled CVT

sizeof(F26Dot6) * (SFAC_LENGTH(ClientInfo, sfnt_controlValue) / sizeof(sfnt_ControlValue))

[PrivateSpaceOffsets->offset_globalGS]

TrueType Global GS

sizeof(fnt_GlobalGraphicStateType)

[PrivateSpaceOffsets->offset_FontProgram]

TrueType Font Program

SFAC_LENGTH(ClientInfo, sfnt_fontProgram)

[PrivateSpaceOffsets->offset_PreProgram]

TrueType Pre Program

SFAC_LENGTH(ClientInfo, sfnt_preProgram)

[PrivateSpaceOffsets->offset_TwilightZone]

TrueType Twilight Element

sizeof(fnt_ElementType)

[PrivateSpaceOffsets->offset_TwilightOutline]

TrueType Twilight Outline

```
fsg_GetOutlineSizeAndOffsets(pMaxProfile->maxTwilightPoints,
                             MAX_TWILIGHT_CONTOURS,
                             &(PrivateSpaceOffsets->TwilightOutlineFieldOffsets),
                             &ulOutlineSize,
                             &ulReusableMarker);
```

sizeof(fsg_OutlineFieldInfo)

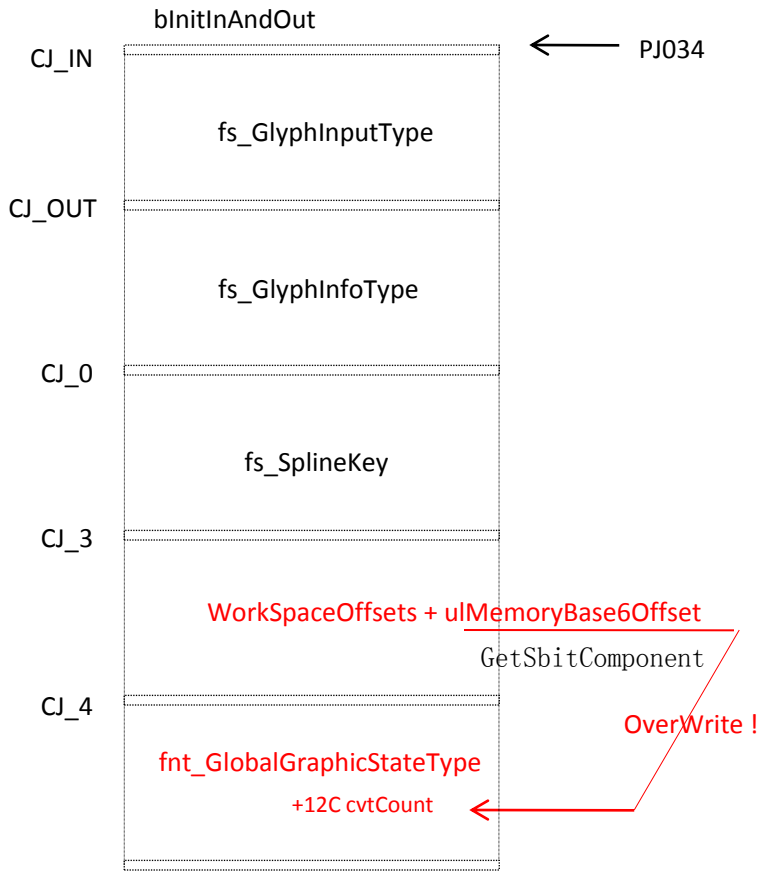
[END]

}

Win32K 模块的调用假设

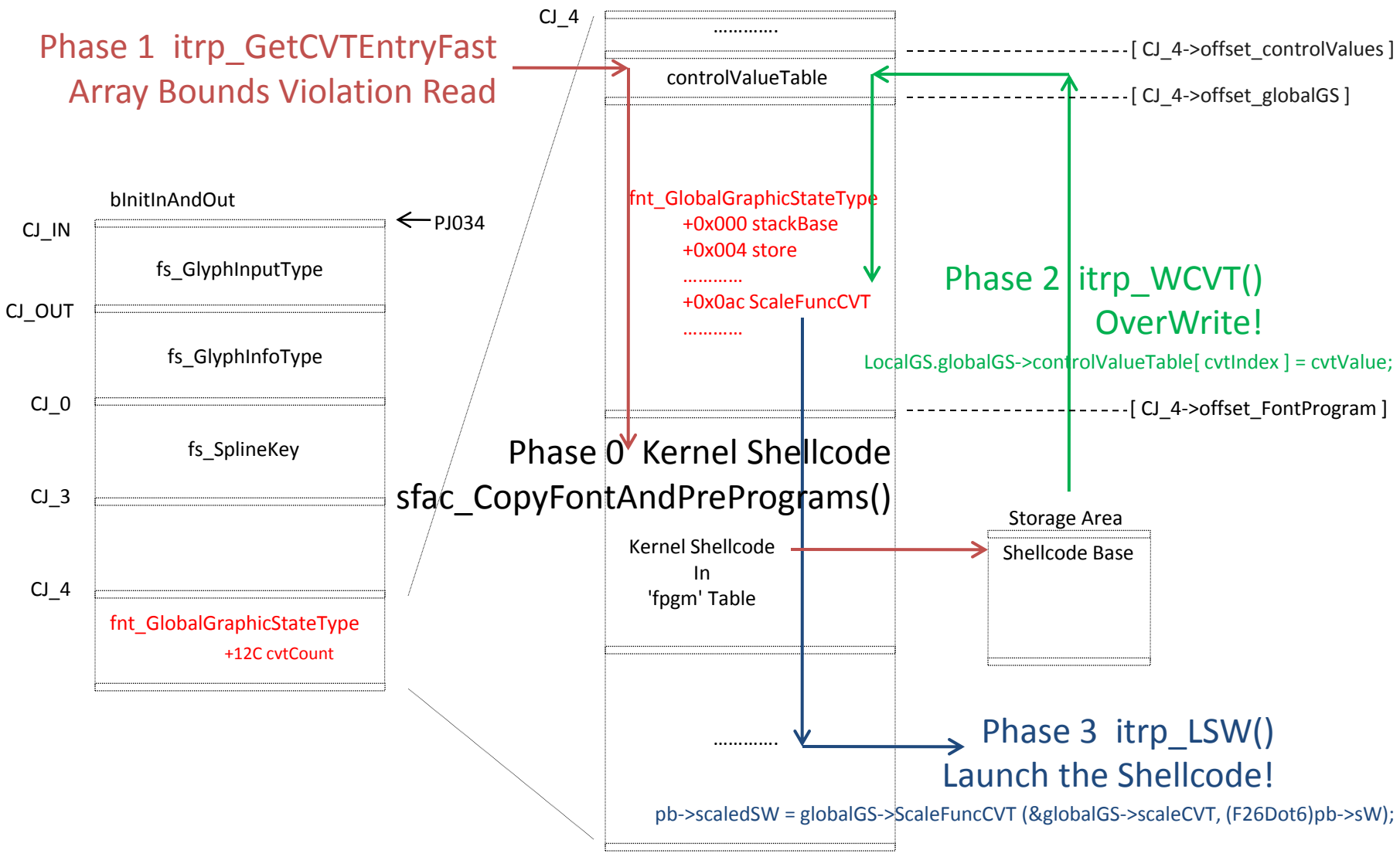
- 堆越界问题？数组越界问题！
- 1）ulReusableMemoryOffset 与 pBitmapPtr2 / pbyRead 机制
- 2）Win32K PJ034 的假定
- 3）GetSbitComponent usXOffset /usYOffset
- 4）sfac_GetSbitBitmap 缺乏越界检测

```
pbyBitMap = pbyBitRow + usXOffBytes;    /* adjust left */

for (usCount = 0; usCount < usSrcRowBytes; usCount++)
{
    *pbyBitMap++ |= *pbyBdat++;
}
pbyBitRow += usDstRowBytes;
usHeight--;
```



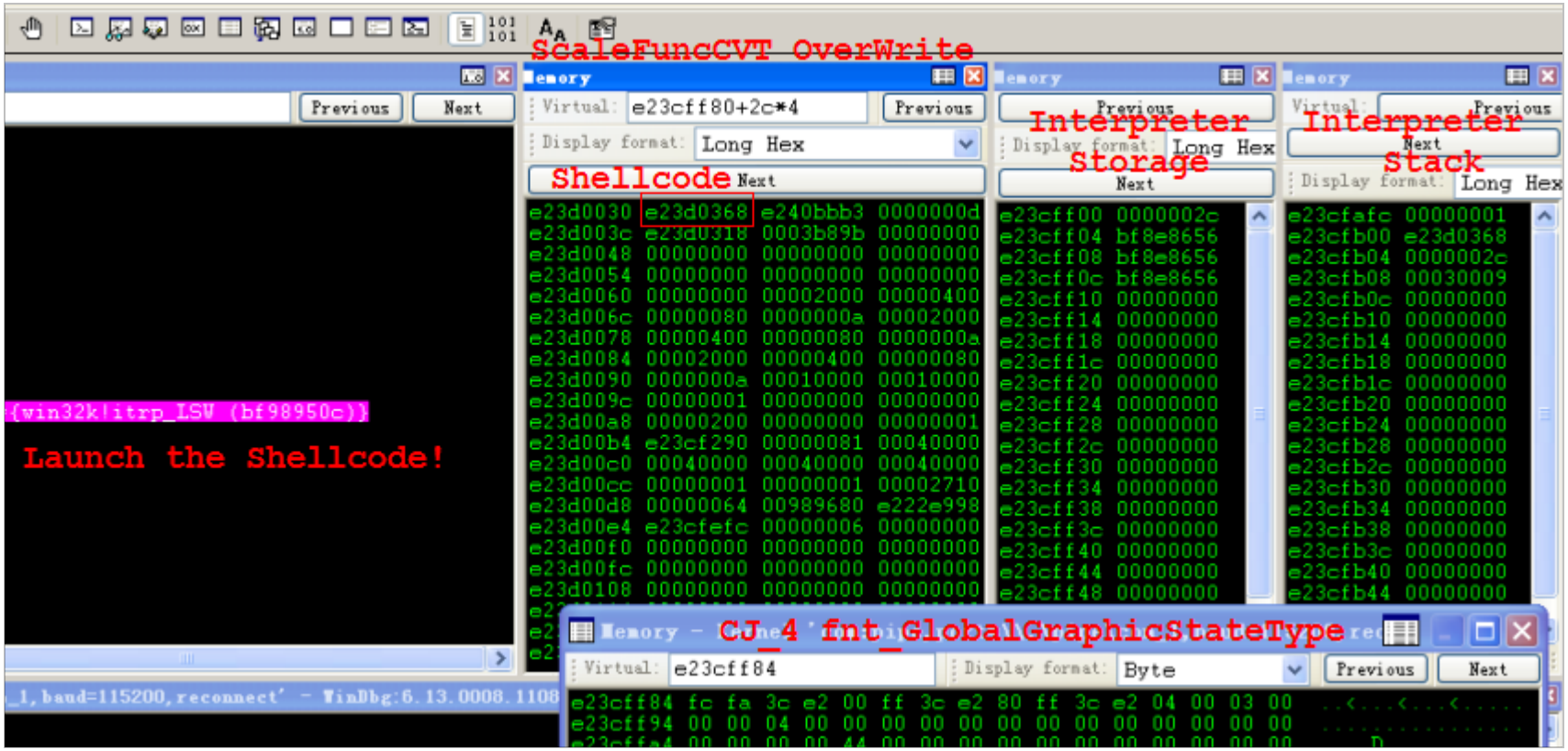
- 这是一个潜藏了二十多年的 Bug

叹为观止的利用技术



叹为观止的利用技术

演示：Duqu 与 MS11-087



第五部分

字体引擎的更多审计

MS11-087 的一个隐含问题

GetSbitComponent 例程的无限递归

演示：Duqu 揭示的另一个潜在问题

```
BUGCHECK_STR:  0x7f_8

eax=00009ed3 ebx=00000008 ecx=e23e6138 edx=e23e0008 esi=e23e6138 edi=e23e6670
eip=bf986328 esp=f5474fe8 ebp=f5475054 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
win32k!sfac_ShaveSbitMetrics+0xd:
bf986328 53              push    ebx
Resetting default scope

DEFAULT_BUCKET_ID:  DRIVER_FAULT

PROCESS_NAME:  csrss.exe

LAST_CONTROL_TRANSFER:  from bf9871ec to bf986328

STACK_TEXT:
f5475054 bf9871ec 0000000a 00000006 00000008 win32k!sfac_ShaveSbitMetrics+0xd
f54750ec bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x19c
f5475190 bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f5475234 bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f54752d8 bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f547537c bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f5475420 bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f54754c4 bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f5475568 bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f547560c bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f54756b0 bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f5475754 bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f54757f8 bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f547589c bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
f5475940 bf987243 e23e0008 0000000a 00000006 win32k!GetSbitComponent+0x1f3
```

MS11-087 的一个隐含问题

GetSbitComponent 补丁细节 —— 引入递归深度参数 a14

```
result = GetSbitComponent(
           v43,
           *(_DWORD *)v6,
           *(_WORD *)(v6 + 44),
           *(_DWORD *)(v6 + 8),
           *(_DWORD *)(v6 + 12),
           v4,
           *(_WORD *)(v6 + 48),
           *(_WORD *)(v6 + 66),
           *(_WORD *)(v6 + 68),
           *(_WORD *)(v6 + 70),
           *(_WORD *)(v6 + 72),
           0,
           0,
           1,
           *(_WORD *)(v6 + 54),
           v7,
           *(_WORD *)(v6 + 94),
           v12,
           v45,
           v9,
           v13);
if ( result )
  return result;
```

```
if ( (unsigned int)(unsigned __int16)a14 + 1 > 20 )
  return 5131;
if ( (_WORD)v22 != 1 && (UIntMult(&v50) < 0 || v50 > a18) )
  return 6656;
a6 = 0;
if ( HIWORD(a17) )
{
  while ( 1 )
```

GetSbitComponent:55

```
ChildEBP RetAddr  Args to Child
f48166c0 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent          1
f4816774 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e    2
f4816828 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e    3
f48168dc bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e    4
f4816990 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e    5
f4816a44 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e    6
f4816af8 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e    7
f4816bac bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e    8
f4816c60 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e    9
f4816d14 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e   10
f4816dc8 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e   11
f4816e7c bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e   12
f4816f30 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e   13
f4816fe4 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e   14
f4817098 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e   15
f481714c bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e   16
f4817200 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e   17
f48172b4 bf98de0c 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e   18
f4817368 bf98de0c e2420008 0000000a 00000006 win32k!GetSbitComponent+0x25e   19
f481741c bf98e63d 00000008 0000000a 00000006 win32k!GetSbitComponent+0x25e   20
f48174a0 bf842f62 e13eb4b0 e13eb4b0 e1f65c90 win32k!sbit_GetBitmap+0x126
f48174e8 bf83e487 e2427010 e2427074 00000001 win32k!fs_ContourScan+0xf3
f481762c bf83d8f9 00000014 00000003 f4817700 win32k!lGetGlyphBitmap+0x181
f4817654 bf83d7e5 00000000 00000001 00000003 win32k!ttfdQueryFontData+0x13e
f48176a0 bf83a7eb e1596010 e1390008 00000001 win32k!ttfdSemQueryFontData+0x45
f48176d0 bf83ab39 e1596010 e1390008 00000001 win32k!PDEVOBJ::QueryFontData+0x3c
f4817748 bf807b05 f4817ab0 e1e4c028 e1f65c90 win32k!xInsertMetricsPlusRFONTOBJ+0x11e
f481777c bf812b48 00000006 f4817b54 7ffdf23a win32k!RFONTOBJ::bGetGlyphMetricsPlus+0x180
f48177b0 bf812614 f4817d38 f4817ab0 00000058 win32k!ESTROBJ::vCharPos_H3+0xee
f48177f4 bf8118ca 7ffdf234 00000006 f4817d38 win32k!ESTROBJ::vInit+0x257
f4817a98 bf813021 f4817d38 00000005 00000005 win32k!GreExtTextOutWLocked+0x666
f4817c00 bf80c6b7 f4817d38 7ffdf1e4 00000060 win32k!GreBatchTextOut+0x344
f4817d54 804de970 0000007a 0012f818 0012f830 win32k!NtGdiFlushUserBatch+0x11b
f4817e0c f788807f 00000000 00000000 00000000 nt!KiFastCallEntry+0xcd
```

只此一处？

EvaluateSpline 例程的递归条件？

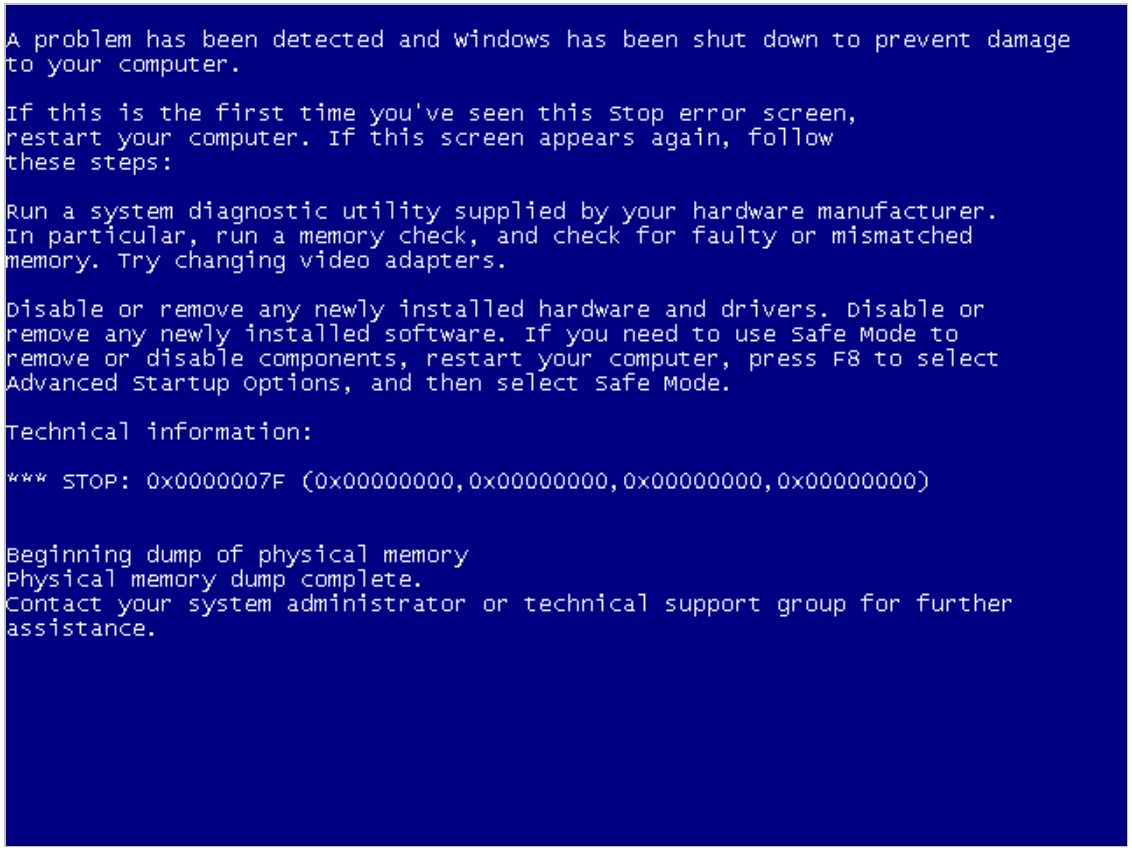
请别忘记引擎现在工作于内核

```
BUGCHECK_STR:  0x7f_8

eax=ec2e3ebf ebx=0000016e ecx=99999996 edx=0000016e esi=0000016e edi=2b3e02af
eip=bf8f37d9 esp=f5b98000 ebp=f5b98034 iopl=0         nv up ei pl nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010216
win32k!EvaluateSpline+0x2:
bf8f37d9 55              push    ebp
Resetting default scope

DEFAULT_BUCKET_ID:  DRIVER_FAULT

PROCESS_NAME:  csrss.exe

LAST_CONTROL_TRANSFER:  from bf8f37a2 to bf8f37d9

STACK_TEXT:
f5b97ffc bf8f37a2 ec2e3ebf 0000016e 0bb620b7 win32k!EvaluateSpline+0x2
f5b98034 bf8f37a2 3ec2e3e8 0000016e 17b92177 win32k!EvaluateSpline+0x1c9
f5b9806c bf8f37a2 e3ec2e3b 0000016e fd728fd3 win32k!EvaluateSpline+0x1c9
f5b980a4 bf8f37a2 2e3ec2e0 0000016e ffb31ff7 win32k!EvaluateSpline+0x1c9
f5b980dc bf8f37a2 c2e3ec2a 0000016e df6b0df2 win32k!EvaluateSpline+0x1c9
f5b98114 bf8f37a2 ec2e3ebf 0000016e 0bb620b7 win32k!EvaluateSpline+0x1c9
f5b9814c bf8f37a2 3ec2e3e8 0000016e 17b92177 win32k!EvaluateSpline+0x1c9
f5b98184 bf8f37a2 e3ec2e3b 0000016e fd728fd3 win32k!EvaluateSpline+0x1c9
f5b981bc bf8f37a2 2e3ec2e0 0000016e ffb31ff7 win32k!EvaluateSpline+0x1c9
f5b981f4 bf8f37a2 c2e3ec2a 0000016e df6b0df2 win32k!EvaluateSpline+0x1c9
f5b9822c bf8f37a2 ec2e3ebf 0000016e 0bb620b7 win32k!EvaluateSpline+0x1c9
f5b98264 bf8f37a2 3ec2e3e8 0000016e 17b92177 win32k!EvaluateSpline+0x1c9
f5b9829c bf8f37a2 e3ec2e3b 0000016e fd728fd3 win32k!EvaluateSpline+0x1c9
f5b982d4 bf8f37a2 2e3ec2e0 0000016e ffb31ff7 win32k!EvaluateSpline+0x1c9
f5b9830c bf8f37a2 c2e3ec2a 0000016e df6b0df2 win32k!EvaluateSpline+0x1c9
f5b98344 bf8f37a2 ec2e3ebf 0000016e 0bb620b7 win32k!EvaluateSpline+0x1c9
f5b9837c bf8f37a2 3ec2e3e8 0000016e 17b92177 win32k!EvaluateSpline+0x1c9
f5b983b4 bf8f37a2 e3ec2e3b 0000016e fd728fd3 win32k!EvaluateSpline+0x1c9
f5b983ec bf8f37a2 2e3ec2e0 0000016e ffb31ff7 win32k!EvaluateSpline+0x1c9
```

更多的审计

那么，又一个潜藏了二十多年的 Bug 喽？

演示



引擎的内核化究竟意味着什么？

WoW！ TOCTTOU！

```
bf86187e 53              push    ebx
bf86187f 51              push    ecx
bf861880 52              push    edx
bf861881 ff5704          call    dword ptr [edi+4]    ds:0023:e16ef13c={win32k!pvGetPointerCallback (bf8e8942)}
bf861884 83c40c          add     esp,0Ch
bf861887 85c0            test    eax,eax
bf861889 0f8413ffffff    je      win32k!sfac_SearchForBitmap+0x3e (bf8617a2)
Command - Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,reconnect' - WinDbg:6.13.0008.1108 X86
1: kd> dd esp
f5b9d460  e145b158 003769d4 000004e8 e16ef4bc
f5b9d470  e16ef4f4 e16ef4ca f5b9d6e8 00000030
```

PS : Rootkit, Object Hook

```
bf86187f 51              push    ecx
bf861880 52              push    edx
bf861881 ff5704          call    dword ptr [edi+4]
bf861884 83c40c          add     esp,0Ch
bf861887 85c0            test    eax,eax
bf861889 0f8413ffffff    je      win32k!sfac_SearchForBitmap+0x3e (bf8617a2)
Command - Kernel 'com:pipe,port=\\.\pipe\com_1,baud=115200,reconnect' - WinDbg:6.13.0008.
1: kd> r
eax=00b169d4 ebx=000004e8 ecx=003769d4 edx=00376ebc esi=e16ef4a0 edi=e16ef138
eip=bf861884 esp=f5b9d460 ebp=f5b9d4a0 iopl=0         nv up ei pl nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000206
win32k!sfac_SearchForBitmap+0x37:
bf861884 83c40c          add     esp,0Ch
1: kd> db 00b169d4 14e8
00b169d4  00 02 00 00 00 00 06-00 00 01 28 00 00 00 a0  ...........(....
00b169e4  00 00 00 05 00 00 00 00-0a fe 0c 01 00 00 00 00  ................
00b169f4  0a fe 00 00 0a fe 0c 00-01 00 00 00 f4 00 00 00  ................
00b16a04  00 62 55 df 0c 0c 01 01-00 00 01 c8 00 00 00 a0  .bU.............
00b16a14  00 00 00 05 00 00 00 00-0c fe 0e 01 00 00 00 00  ................
```

第六部分

尾声

沉思

- 误解

"One popular internet assumption about the Duqu exploit was its dependency on a new vulnerability in Microsoft Word's parsing of the OLE2 document format and allowing activation of the CVE-2011-3402 exploit in the kernel. In turn, the newer .docx format was considered to be more secure"...

— Anatomy of Duqu exploit, vb100

沉思

- 罗生门

"Moving ... the GDI from user mode to kernel mode has provided improved performance without any significant decrease in system stability or reliability"...

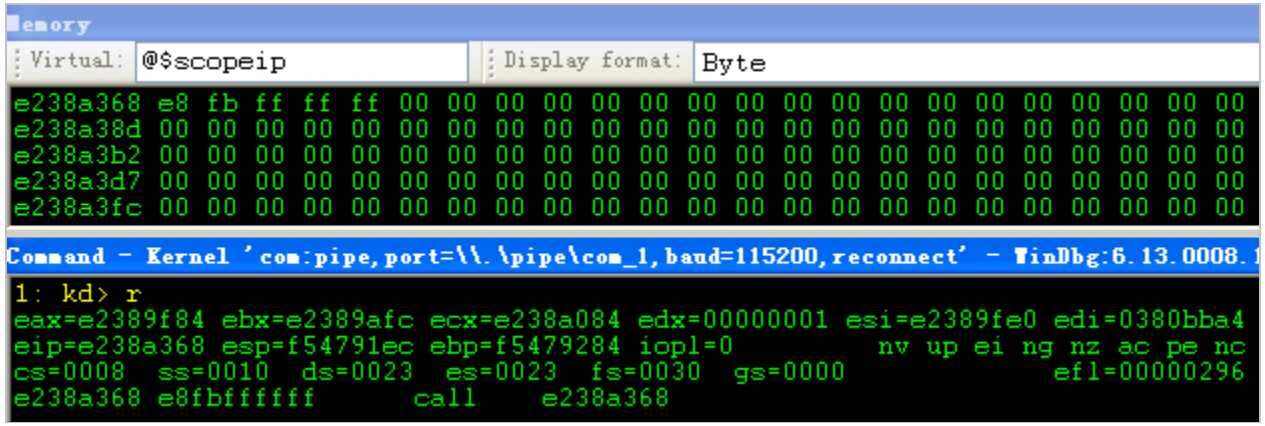
— Windows Internals, 4th Edition

"GDI represents a significant kernel attack surface, and is perhaps the most easily accessible remotely. This resulted in perhaps our most critical discovery, remote ring0 code execution when a user visits a hostile website (even for unprivileged or protected mode users)."...

— There's a Party at Ring0 and You're Invited, bh2010

沉思

- SMEP ？ Kernel Mode Shellcode



这是一个时代的产物，也许它应该进博物馆了。

致谢！

P1P1Winner　PJF　Bugvuln　RoyceLu

YaoTong　PaulFan　MJ0011

360-deepscan-team　360-hips-team

SyScan Committee　Fan.Tuan

Q&A

wangyu@360.cn