应用软件缺陷利用的一点心得(Webkit 篇)

By wushi

众所周知，在各软件厂商高度重视软件安全的当前情况下，成功稳定的利用软件的缺陷进行娱乐是一件越来
越困难的事情。

从这篇文章
(http://blogs.technet.com/srd/archive/2009/08/04/preventing-the-exploitation-of-user-mode-heap-corruption-vulnerabilities.aspx)
我们可以看到，"邪恶帝国"对于软件的保护，设置缺陷利用的障碍已经是"令人发指"，无所不用其计。在后
XP 时代，想要攻击采用系统内存管理的软件几乎是不可能的事情。但是所幸的是，很多软件厂商出于种种原因，
很重要的一个原因是出于对不同平台移植性的考虑，没有采用 windows 的系统内存管理。
这里一个重要的例子就是 MS office,它为了在 MAC OS 下可以方便移植，也没有采用 windows 的系统内存管理，
这些软件在可预见的将来，估计也不会采用 windows 的系统内存管理，这就给了我们一个生存空间，毕竟，
这些软件的保护措施一般不会像 windows 这样严格。

在这些软件中，我们应该把注意力集中在支持 script 的软件上。让 script 在软件中执行，就意味着很大程度
上我们可以控制内存的分配，让合适的内容在合适的时间出现在合适的地点。

言归正传，我们下面来看看 webkit 的情况. webkit 是 safari 和 chrome 的核心，其中一个函数是这样的：

```
110
111 void RenderArena::free(size_t size, void* ptr)
112 {
113 #ifndef NDEBUG
114 // Use standard free so that memory debugging tools work.
115 RenderArenaDebugHeader* header =
static_cast<renderarenadebugheader*>(ptr) - 1;
116 ASSERT(header->signature == signature);
117 ASSERT(header->size == size);
118 ASSERT(header->arena == this);
119 header->signature = signatureDead;
120 ::free(header);
121 #else
122 // Ensure we have correct alignment for pointers. Important for Tru64
123 size = ROUNDUP(size, sizeof(void*));
```

```
124
125 // See if it's a size that we recycle
126 if (size < gMaxRecycledSize) {
127 const int index = size >> 2;
128 void* currentTop = m_recyclers[index];
129 m_recyclers[index] = ptr;
130 *((void**)ptr) = currentTop;
131 }
132 #endif
133 }
```

这个函数是 webkit 的 render object 的内存管理的一部分，所谓 render object 就是显示在网页上的 table 啊，
图片啊这些东西。这些东西可以通过 javascript 用 DOM 的方式操作。目前我研究的大部分浏览器问题是出在
这个 DOM 上，特别是 use after free 的问题。纠其原因，在于浏览器在处理 DOM 对象的异步操作上，由于 JS 的
灵活性，程序员无法考虑到所有的情况，那么出现错误就在所难免了。

但是 use after free 这类问题，稳定的利用往往是一个很大的问题，一段内存在释放之后，其内容可能是
任意值，如何把这任意值变成我们需要的东西是一件比较费力气的事。从这个 free 函数我们可以看出，呵呵，
没有 MS 的那些乱七八糟的保护措施，对于我们来说是一个好消息，这个函数非常简单，把释放的内存放到
对应的链表（按释放内存的大小分为不同的链表）里，修改链表的头为被释放的指针，指向前一个链表头.
这个内存管理对于攻击者来说简直是天堂，对于一个 overflow 类型的缺陷可以非常轻松的覆盖一个另一个
obj 的 vtable,从而实施攻击。

对于未初始化和 use after free 类的缺陷，情况稍微复杂一点，但也不是什么难事。
以 use after free 为例，一个 render object free 的时候，从上面的算法可知，ptr+0 的位置会填入前一个
释放指针的地址，而 ptr+0 在 c++的 object 里就是 vtable 的所在，所以刚释放的这个 object 的 vtable 会填入
前一个释放的 object 的地址。

当调用刚释放的 object 的函数时，这个函数地址事实上是指向前一个释放的 object 的某个属性的位置，如果
我们能够控制这个属性的值的话，我们就可以任意控制 EIP。

非常幸运的是，这个属性在大多数情况下是能控制的，由于一个 render object 的很多属性是由 CSS 定义的，
CSS 属性很多是数字型的，比如 bottom,left,right,top 等等，我们定义好这些值就能控制 eip 了，哈哈。

这个方法应用起来是这样的:在 JS 中定义两个同类型的 render object,首先释放一个 CSS 经过精心设计的 object,
然后释放第二个,接着做一个导致第二个产生 use after free 的操作.

大家不禁要问,这么费事,直接用 heap spraying 得了.但是 webkit 的 string 是非常不同的,叫 ustring,即 unicode string,
我们来看看一个具体的例子.

```
0:000> dd 7fd9b4b0
7fd9b4b0 00000000 0000002d 00000001 70e0ffd1
7fd9b4c0 7ff130f2 00000000 7fef44e0 00000000
7fd9b4d0 00000000 0000002d 0000002d 00000000
```

7fd9b4b0+0 是 offset,"0000002d"是 ustring 的长度, "00000001 "是这个 ustring object 的 reference count,
"70e0ffd1"是这个 ustring 的 hash 值,"7fef44e0 "里才放着这个 ustring 的具体内容。

"7fef44e0"是 fastMalloc 出来的东西,和这个 ustring object 是采用不同的内存管理方法,我几乎可以肯定的说,
直接用 string 的内容来做 heap spraying 没有任何作用,因为 string 的内容在内存中几乎可以看作是另一个 heap 的
(我说的含糊一点,哈哈,在 chrome 下是另一个 heap 的,在 safari 下由于没有 symbols, 我不敢肯定)。

所以 string 内容在 object 发生问题时是毫无用处的,但是能不能用 heap spray 的方法呢?哈哈,是可以的,但是要变通一下,
先看看上面这个 ustring object 变成汇编语言是什么东西?

```
0:000> u 7fd9b4b0
7fd9b4b0 0000 add byte ptr [eax],al
7fd9b4b2 0000 add byte ptr [eax],al
7fd9b4b4 2d00000001 sub eax,1000000
7fd9b4b9 0000 add byte ptr [eax],al
7fd9b4bb 00d1 add cl,dl
7fd9b4bd ffe0 jmp eax
```

haha,我们保证 eax 是这个 heap 里的地址,构造一个长度为 0x2d 的字符串,控制一下 reference count 的值,然后构造
string 的内容,让它的 hash 值中间 2byte 为 0xffe0(jmp eax),我们就可以跳到另一个较远的地址去.我们在处理
object 的 heap 中用 javascript 大量的生成这样的对象,程序出问题时就有较大可能执行上面这些指令.

再用 heap spraying 保证整个内存空间中写入了大量的 shellcode,这样我们基本就可以成功的利用了.
一个长度为 0x2d,hash 值为"70e0ffd1"的 string 如下:

```
addr4="\u543d\u4044\u3a7a\u4361\u5977\u696c\u2566\u4151\u5371\u275e\
u4c48\u5252\u5b38\u4c44\u742d\u5827\u6a7a\u6644\u2647\u4e4a\u6565\u
6825\u332e\u232d\u7456\u406d\u6630\u6841\u524c\u2955\u242b\u3c21\u4
628\u3e50\u687d\u7e58\u313d\u6653\u3e2c\u3468\u2d42\u464a\u7361\u54
30\u3051";
```

这个方法我说的比较概括,为了避免麻烦,我也不打算附上一个具体的例子.但是你有志写一个 safari 的 POC 时,
可以参考参考,哈哈.

我举这个例子的意思是:即使是在 win7&vista 时代,我们依然能够通过具体问题具体分析的方法,绕过种种限制,
让软件缺陷成为真正的威胁.