Exploit IE Using Scriptable ActiveX Controls

趋势科技中国研发中心

古河

1.背景介绍

当时之所以会做这个东西，是因为写一个 exploit，需要过 EMET。大家都知道 EMET 里面的 EAF 很烦人，旧的 shellcode 会被检测到，然后我人有太懒，不想重新写 shellcode。于是就有了这样一个想法：能不能不通过 shellcode，也能实现 exploit 之后的 payload 功能呢？

所以我们的目标如下：

1. 首先要有个漏洞，可以造成内存修改（理想的状况是转化成了任意地址读写）
2. 不使用任何 shellcode
3. 不使用 ROP、VirtualProtect、NtContinue 之类的需要控制 EIP 的技术
4. 要有一定的通用性

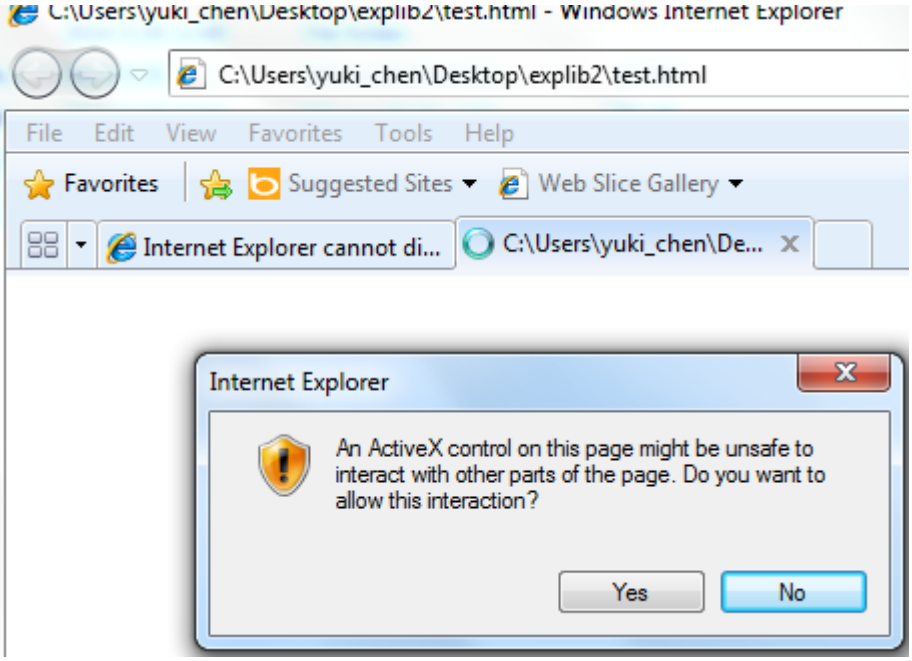
而我找到的满足以上条件的方案是：**使用浏览器所支持的脚本（JavaScript 或者 VbScript），来实现 payload 的功能。**

2. 浏览器脚本和 ActiveX Control

订好方向以后，我选择了 JavaScript 作为目标脚本语言（事实证明这个选择是错误的，带来了一些不必要的麻烦，这个后面会提及）。
大家知道，IE 里面是可以通过脚本来调用一些 ActiveX (COM)组件的，比如下面的 JavaScript 脚本可以创建一个计算器进程：

```
var WshShell = new ActiveXObject("WScript.shell");
oExec = WshShell.Exec('calc.exe')
```

但是由于 IE 的安全设置，直接在网页里面嵌入上述脚本是行不通的，会弹提示框或者干脆调用失败：



那么我们的目标就很明确了：在可以读写内存的情况下，能不能找到一个办法，能够绕过这些安全设置，让脚本得以顺利执行呢？答案是肯定的。

3.SafeMode 标志位

经过一番调试逆向，我们发现 jscript(jscript9)的相关代码里面，有一个字节的标志位是可以控制这些安全设置的。在我们试过的各个版本（IE8、10 和 11）中，当满足 flag & 0xB == 0 的时候，就可以绕过安全设置执行危险动作而无任何提示。

这个标志位在各个版本的 IE 中略有不同，然后不同的 jscript 小版本可能也有差异：
在 jscript (5.8.7601.17866)中，是 COleScript+0x188

```
; public: int __thiscall COleScript::CanObjectRun(s
?CanObjectRun@COleScript@@QAEHABU_GUID@@PAUIUnknown
                                        ; CODE XREF

var_4           = dword ptr -4
arg_0           = dword ptr  8

                mov     edi, edi
                push    ebp
                mov     ebp, esp
                sub     esp, 44h
                mov     eax, ___security_cookie
                xor     eax, ebp
                mov     [ebp+var_4], eax
                push    ebx
                mov     ebx, ecx
                mov     eax, [ebx+188h]
                push    esi
                mov     esi, [ebp+arg_0]
                push    edi
                mov     edi, edx
                test    al, 0Bh
```

在 jscript9 (11.0.9600.16518)中，是 ScriptEngine+0x1F0:

```
long __thiscall ScriptEngine::GetSafet
                                ; CODE
                                ; Scrip

        = dword ptr -28h
        = byte ptr -24h
        = dword ptr -4
        = dword ptr  8

        mov     edi, edi
        push    ebp
        mov     ebp, esp
        sub     esp, 28h
        mov     eax, ___security_cookie
        xor     eax, ebp
        mov     [ebp+field_2C], eax
        push    esi
        mov     esi, ecx
        push    edi
        mov     edi, [ebp+field_38]
        push    ecx                     ; int
        mov     eax, [esi+1F0h]
```

我们要做的就是在实现任意地址读写（或者至少在特定条件下往可控的相对地址写 0）之后，将这个对应字节设为 0

以 IE11 (jscirpt9.dll: 11.0.9600.16518)为例，我们可以通过如下两步得到 ScriptEngine 的地址：

```
var func_addr = this.leakAddress(ActiveXObject);
var script_engine_addr = this.read32(this.read32(func_addr + 0x1c) + 4);
```

然后就随便怎么玩了

4.叫你不用 VBScript

对于 IE11 之前的版本来说，直接将标志位设成 0 就大功告成了。但是 IE11 里面会有额外的检测，具体做法是：在内部每次设置标志位时，都会带上这个标志位去计算一个 hash 值 h1，每次使用标志位之前重新计算这个 hash 值 h2，并比较前后两个 hash 是否相等，如果不等就会 crash。也就是说，我们在 IE11 里面，如果直接修改这个标志位，会无法通过系统的检测。这里不得不赞一下微软在安全方面下的功夫，显然他们设计是还是考虑到了要保护这些薄弱环节。

但是 IE11 里的保护还是有办法绕过的，具体方法就不再这里讲了，大家可以看我给出的示

例代码。

费了很大力气绕过 IE11 检测，但是后来经过瀚海源的高富帅壕哥的提醒，我发现 IE11 的 vbscript.dll 里面是没有类似检测的，也就是说，直接用 VBScript 就行了！我 XO@%^*&^%(

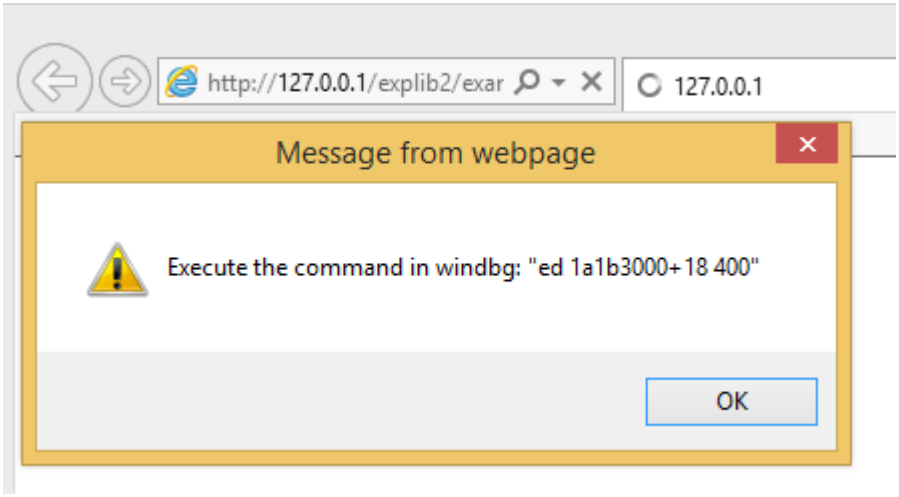
5.示例代码

示例代码我已经上传，大家可以看看：

https://github.com/guhe120/explib2

另外这份代码的环境是 IE11 + Windows8.1，如果用其它环境，可能会造成一些结构偏移不同而提前 crash。
简单讲一下测试步骤：

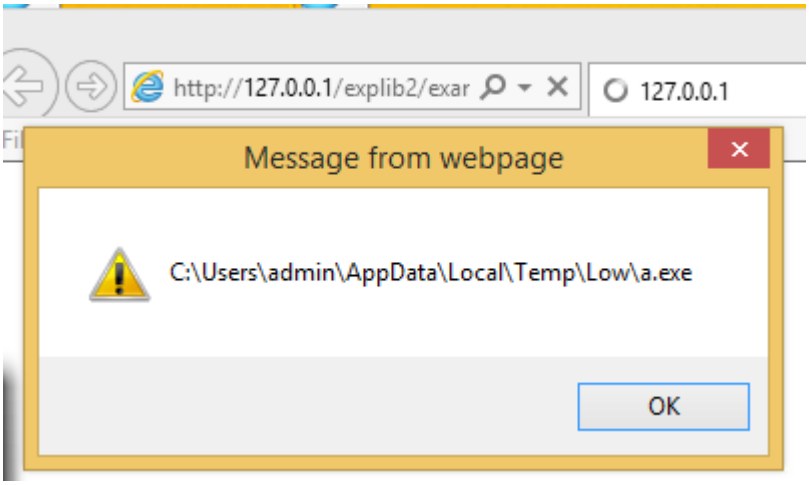
1. 首先把 explib2 里面的所有内容拷贝到你的服务器目录下面
2. 访问 explib2/example.html



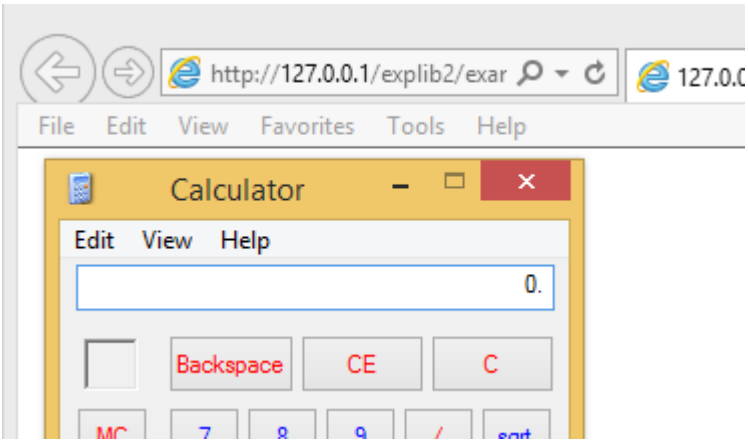
3. Attach windbg，修改内存来模拟数组长度破坏:
   ed 1a1b3018 400
   .detach

   然后点击对话框继续

4. 这里会提示我们 drop 了一个 exe 下来（以前是用 shellcode 来做的）



5. 人民群众的好朋友计算器



之里要注意的是新起的计算器在某人配置下是 Low Integrity 的，使用脚本我们也无法突破 IE11 的沙盒机制。

6.一些杂七杂八

个人觉得这中在实现任意读写的情况下的一些技巧，玩玩还可以，但是实际效用不见得太高。反而是前面的步骤诸如 UAF 如何转换成内存读写啊，64 如何稳定搞啊，已经后面的像是如何突破沙盒之类的，似乎更为重要一些。

我不太清楚这篇文章里用到的技术，和 TK 教主价值 5w$的马赛克、或者 YuanGe 的 DVE 技术是否有相同的地方。但是我可以肯定的是这个东西一定早就有许多人已经知道，比如瀚海源的同学。所以这里由我写出来其实显得有点厚脸皮，或者说不知天高地厚。这里相对各位前辈高人打个招呼，拙文如有冒犯之处，还望勿怪。

7.结语

可能是因为在夜深人静的时候写这篇文章，写着写着突然莫名地有些感伤。
遥想五年前的今天，自己刚刚走出校门，意气风发；
HR 问起职业规划，竟大言不惭地说，要用五年时间，成为这个领域的专家；
五年过去，终于明白了自己无非只是一只井底之蛙；
而除了工作，也有太多的事情变得需要去牵挂；
所幸还是愿意坚持当初的选择，重整出发；
坚信明天的自己会比今天更棒