Exploit IE Using Scriptable ActiveX Controls

Yuki Chen (aka guhe120)

1. Background

Several months ago, I tried to write a poc exploit for Internet Explorer 11 with one of our vulnerabilities. One of the goals was to make this poc exploit bypass microsoft's EMET tool. As you may know, there is a feature called EAF (Export Address Filtering) in EMET, which tries to detect suspicious shellcode behavior by monitoring the access to certain system modules' export tables. While this feature is not so hard to bypass, we still need to rewrite our old shellcode to overcome it. Since I'm too lazy to do the shellcode refracting job, I began to think about one question: Is it possible to achieve the exploit and the payload behaviors without using any shellcode?

So below are our assumptions and rules:

1. First, we have a vulnerability to achieve memory leak or corruption (it would be better if it can be converted to arbitrary memory read/write).
2. Using of any kind of shellcode is not allowed.
3. Using of ROP、VirtualProtect、NtContinue is not allowed.
4. It can bypass all features in the latest version of EMET.
5. It should work for different IE versions (not only IE 11).

And our solution is: **Use the browser supported scripts (JavaScript or VBScript) instead of shellcode to achieve the payload behaviors**.

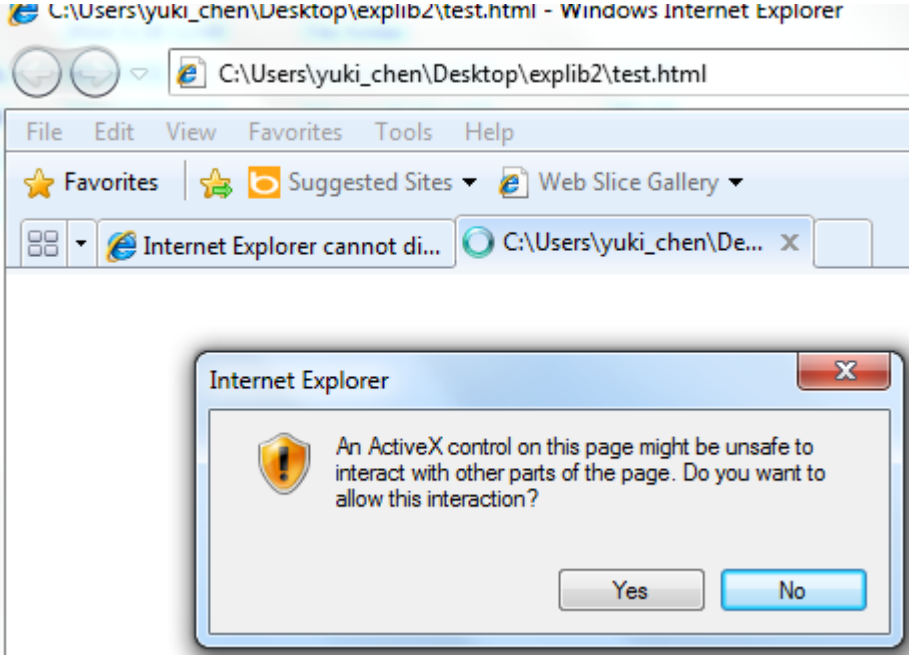
2. Browser Scripts and ActiveX Controls

I selected Javascript as our target script language. As you may know, we can use javascript to call some ActiveX (COM) controls, for example, the following javascript code will launch a calculator:

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

However, if you embed the above script in a html and open it with IE, you won't success because of the security settings of Internet Explorer (You may get a warning dialogue or just fail directly based on your security setting):

:



Then our task is clear: After we achieved memory read/write with our vulnerability, we need to find a way to bypass the security settings for using ActiveX controls. Is it possible? My answer is YES.

3. The Safe Mode Flags

After some reversing & debugging in IE, we found that there is a flag in jscript9.dll (or jscript.dll for IE8) which is used to control the security setting. In all IE versions which we've tested on (IE8、10 and 11)，if the condition "flag & 0xB == 0" is met, we will be able to call the unsafe ActiveX controls using javascript code, without any alert!

Below are some places where the flag exits：
In jscript (5.8.7601.17866)，it's in COleScript+0x188

```
; public: int __thiscall COleScript::CanObjectRun(
?CanObjectRun@COleScript@@QAEHABU_GUID@@PAUIUnknow
                                        ; CODE XRE

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
```

And in jscript9 (11.0.9600.16518)，it becomes 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]
```

What we need to do it to set this flag to 0, then our browser will run in god mode.

Take IE11 (jscirpt9.dll: 11.0.9600.16518) as an example，by leaking certain objects and their data, we are able to get the address of ScriptEngine：

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

4. Seriously, I should try VBScript first

For IE versions before IE11, we just need to simply set the flag to 0 and we win. But in IE11, Microsoft has added extra checks on this flag. When the flag is set by internal code, a hash “h1” is computed based on the value of the flag, the address of certain objects and some random data (And the functions EncoderPointer/DecodePointer are also used to make it harder to predicate/crack the hash value). After that, when the flag need to be used in some security check, another hash “h2” will be computed using the same algorithm. And if “h2” is not equal to “h1” (In condition that we directly modify the value of the flag), the check will fail and IE will exit.

However we are still able to bypass this check in IE11, you can refer to my demo code to find out how we do it.

Later when we take a look into the vbscript, we are surprised that vbscript.dll in IE11 does not contain the same check on the flag! So if we choose VBScript at the very beginning, we can save a lot of efforts. What the **** !

Another solution is that you can force IE to load “jscript.dll” other than “jscript9.dll”.

6.The Demo Code

You can find our demo code here：

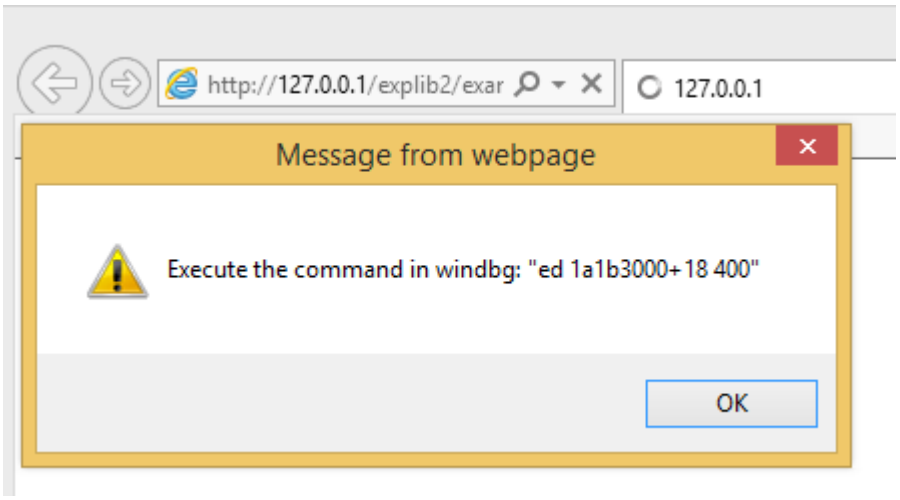
https://github.com/guhe120/explib2

It’s import that before you start to try the demo code, you should make sure that you environment is OK. We developed the demo code on Windows 8.1 32bits and IE 11 32bits, the following dll versions may be importatnt for the demo to success:
Javascript9.dll (11.0.9600.16521)
msado15.dll (6.3.9600.16384)

If you use other versions, you might get a crash and you will need to fix some offsets according to your version.

To test the demo code:

1. Copy the folder “explib2” to your web server directory.
2. Open IE and visit the page: explib2/example.html

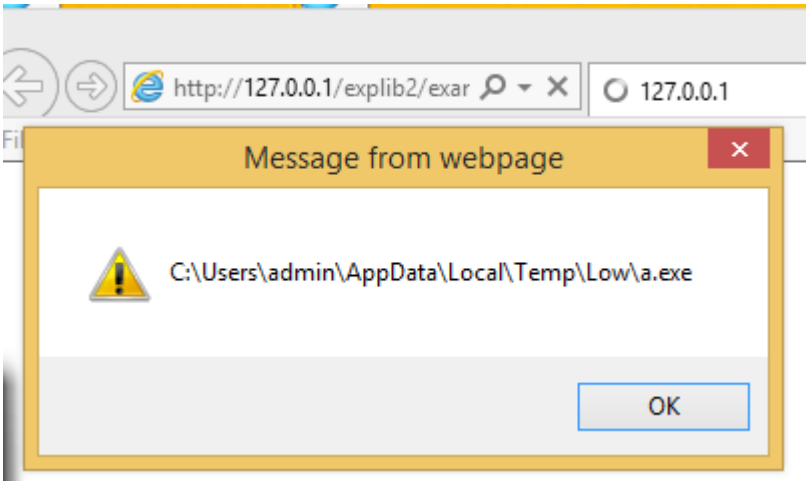


3. Attach windbg，execute the following command, which is used to simulate the condition of a javascript array with corrupted length (In our poc exploit, we achieved this by using our vulnerability)

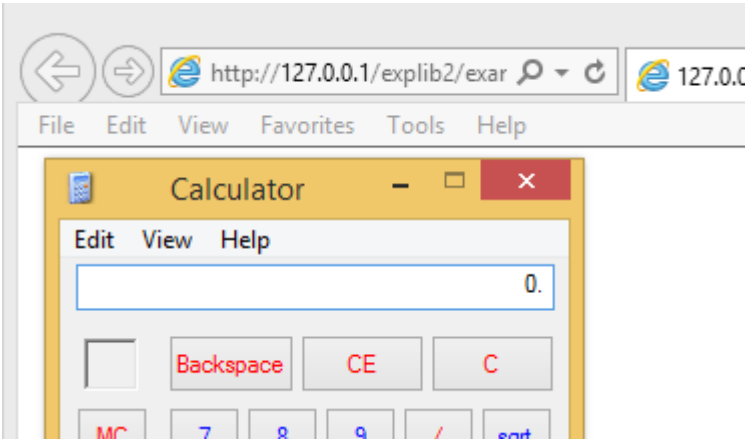
   ed 1a1b3018 400
   .detach

   Click the dialogue box to continue

4. Here is our debug information to notify us that we have dropped an exe file in the temp folder using javascript（which is usually achieved by shellcode）



5. Finally our old friend pops up:



Note that since IE11 runs in Low Integrity Level by default in Windows 8.1, we are only able to create low integrity child process.