doi 10.3969/j.issn.1671-1122.2014.06.007

一种非堆喷射的 IE 浏览器漏洞利用技术研究

宁戈，张涛，文伟平，梅瑞

（北京大学软件与微电子学院，北京 102600）

摘　要：随着互联网技术的进步和发展，计算机成为人们日常生产生活不可缺少的工具，计算机系统的安全问题愈加重要。目前，利用各类系统或软件的漏洞已经成为主流的攻击方法。为更加有效防御针对漏洞的攻击，就需要对各类漏洞利用方法深入研究。文章基于流行的 IE 浏览器漏洞利用方法的研究，介绍了一种新型的浏览器漏洞利用技术，并在已知漏洞中得到了验证。

关键词：IE 浏览器；堆喷射漏洞；漏洞利用

中图分类号：TP309　　文献标识码：A　　文章编号：1671-1122（2014）06-0039-04

Study of Non-Heapspray IE's Vulnerability Exploitation Technique

NING Ge, ZHANG Tao, WEN Wei-ping, MEI Rui

(*School of Software & Microelectronics, Peking University, Beijing* 102600, *China*)

Abstract: With the progress and development of Internet technology, the computer has been the indispensable tool in people's daily life. The security issue of computer system becomes increasingly significant. At present, vulnerability exploitation of systems or software has become a popular attacking method. In order to defend the attack to vulnerability more effectively, we need to study various methods of vulnerability exploitation. This paper introduces a new technique of browser's vulnerability-exploitation, which has been verified in the known vulnerabilities, based on popular methods of IE's vulnerability exploitation.

Key words: IE browser; heapspray vulnerability; vulnerability exploitation

0 引言

随着信息技术的进步，计算机已经深入到社会生产生活的方方面面。大到科学研究、工业及军事设备的实时控制、信息数据的管理，小到日常办公、个人娱乐，无一能离开计算机。互联网的出现和发展，给计算机技术提供了新的发展契机。社交网络、数据存储、网上银行等，让人们越来越体会到了计算机及互联网世界带来的便捷。然而事物总是有两面性的，互联网带来便捷的同时，也不得不让我们去考虑安全问题。

浏览器是用户与互联网的接口，是不可或缺的计算机软件。由于浏览器软件的重要性，利用浏览器漏洞对系统进行攻击已经成为 APT 攻击中重要的一部分。一方面，针对浏览器的安全技术，如针对浏览器漏洞的 Fuzz 方法、浏览器类漏洞的利用技巧也逐渐趋向于成熟。另一方面，从防护角度看，如 IE 浏览器中逐渐加入了沙箱机制、ForceASLR、vtguard 等，以对抗各种漏洞利用技术 [1-4]。

在传统的浏览器漏洞利用过程中，堆喷射技术是重要的一环 [5]。堆喷射可以完成下面几个功能：

1）在堆上伪造虚函数表。

2）在堆上布置 SHELLCODE。

收稿日期：2014-05-13

基金项目：国家自然科学基金 [61170282]

作者简介：宁戈（1988-），男，山西，硕士研究生，主要研究方向：系统与网路安全、漏洞分析及利用技术；张涛（1987-），男，江西，硕士研究生，主要研究方向：系统与网络安全、软件安全漏洞分析；文伟平（1976-），男，湖南，副教授，博士，主要研究方向：网络攻击与防范、恶意代码研究、信息系统逆向工程和可信计算技术等；梅瑞（1984-），男，安徽，硕士研究生，主要研究方向：系统与网络安全、软件安全漏洞分析、信息系统逆向工程等。

3）1）和 2）可以稳定地布局在某堆地址上。

堆喷射有其自身的缺陷，喷射过程中会消耗大量内存，在这一过程中，浏览器会出现卡顿现象，漏洞利用效果较差。而且在 IE9 以后的版本中，浏览器引入了 Nozzel 机制限制了字符串喷射的方法 [6]。因此一种非堆喷射的漏洞利用技术研究是很有必要的。本文对 Peter Vreugdenhil 在 CVE-2012-4792 的漏洞利用方法进行了分析，并验证了其非堆喷射利用方法的可行性。

1 浏览器类漏洞利用技术

1.1 漏洞利用整体思路

如今的浏览器类漏洞主要有两种：堆溢出漏洞和 UAF（use-after-free）漏洞。由于堆类漏洞，在堆上的关键数据有对象的虚函数表指针。如果能够改写虚函数表指针，就能控制程序流程。而要达到改写虚函数表的目的，就需要漏洞能完成任意地址写。对于堆溢出漏洞，其特点就是越界写，因此基本可以达到任意地址写的目的。漏洞满足任意地址写后，需要合理布局堆内存，修改关键数据结构，控制程序流程 [7-9]。

但是对于 UAF 类漏洞，首先要确定被释放的内存是否可以重新控制。其次，确定可以控制后，又分两种情况，一种是可以直接通过可控数据伪造对象，并在对象中写入伪造的虚函数表指针，后面所描述的 CVE-2012-4792 就是这种情况；另一种需要借助程序后面的执行流程，间接完成任意地址写，这种需要合理的堆空间布局，将关键对象布局在某处，最终达到修改关键对象虚函数表指针目的。

无论是堆溢出漏洞或是 UAF 类漏洞，成功伪造虚函数表指针后，需要在内存中放置伪造的虚函数表，传统的漏洞利用技术通常通过堆喷射来完成。如果忽略绕过保护机制，到此基本完成了对于浏览器漏洞的利用。

由于现代操作系统的安全机制，劫持程序 EIP 后，还需要绕过操作系统的各类安全机制。在浏览器内，最关键的是绕过 ASLR 和 DEP 保护机制。现在主流绕过 DEP 的方法是通过构造 ROP 链，调用某些 API 函数，修改内存属性或是关闭 DEP[10,11]。对于 ASLR 的绕过，早期是通过未加入链接选项 /DINAMICLINK 的动态链接库达到目的，如 jre 中的 msvcr.dll，或者是 .Net 中的某些库。如今较为常见的绕过 ASLR 技术是通过泄露某模块的基地址，达到绕过目的 [12]。其他一些安全研究员也提出了各种可能的绕过方法，如 Dion Blazakis 提出的 JIT Spray 和绿盟科技的安全研究员余旸提出的利用 SharedUserData 的技术 [13,14]。值得一提的是，在新型漏洞利用思路中，著名安全研究员 yuange 提出了 DVE 技术，不但无需堆喷，完全绕过现有防护方法，而且可以简化 SHELLCODE 编写。若该技术成熟，则将是漏洞利用技术上的一个里程碑。

1.2 传统堆喷射思路

传统堆喷射通过构造 BSTR 字符串完成。大量的 BSTR 字符串分配可以完成内存占位。通过字符串进行堆喷射有两点优势：一是字符串长度可控，意味着喷射堆块的大小可以控制；二是由于字符串内存可以控制，因此喷射内容可以控制。在很长一段时间，通过 BSTR 完成堆喷射一直是浏览器漏洞利用的第一选择。具体堆喷射代码如代码 1 所示。

代码 1

```
function heap_spray()
        {
shellcode=unescape("%u9090%u9090%u9090%u9090");
            chunk_size = 0x40000;
nopsled = unescape("%u0c0c%u0c0c");
nopsled_len = chunk_size - shellcode.length;
while (nopsled.length < nopsled_len)
   nopsled += nopsled;
nopsled = nopsled.substring(0, nopsled_len);
code = nopsled  + shellcode;
heap_chunks = new Array();
for (i = 0 ; i < 1000 ; i++)
{
   heap_chunks[i] = code.substring(0, code.length);
}
        }
```

在浏览器中，直接赋值某字符串并不能在浏览器内存空间中分配内存。真正触发内存分配的 js 代码为 substring() 函数。堆喷射代码 1 中，通过 substring() 函数控制了每个喷射堆块的大小，同时在浏览器内存中分配了大量内存块，以达到布局 SHELLCODE 目的。在 IE9 以后的版本中，通过 BSTR 字符串喷射内存已经不再可能。但是依旧有方法大量喷射内存，如通过对象属性喷射、喷射内容通过属性值控制，还有如数组喷射或 html5 的 canvas 对象喷射。因此内存喷射方法可以有多种形式，只要能使内存块在某些稳定地址占位，就可以达到目的。堆喷射内存如图 1 所示。

图1 堆喷射内存

2 非堆喷射浏览器漏洞利用技术

因为堆喷射有着天然的缺陷，分配大量内存势必需要一定的时间。因此在漏洞利用过程中，浏览器可能会出现卡顿等现象，某些极端情况下，会导致漏洞利用不成功。而且在微软的 EMET 或是新版本的浏览器中在防护堆喷射方面做了一定改进。这样如果采用某种方法，在整个漏洞利用过程中，不使用堆喷技术，将会大大提高漏洞利用效率。

在浏览器 UAF 类漏洞利用过程中，第一步就是使被释放的内存块重新分配内存并且内存中的数据可控。之后浏览器在继续执行代码的过程中，会错误重用伪造的数据，如寻址到错误的虚函数表地址中，调用错误的虚函数地址。一般的利用过程会通过字符串伪造对象，并且字符串内容精心构造，大小和被释放的内存块大小相当。这样当 LFH 激活后，浏览器一定会在释放的内存块中重新申请内存，内存中的内容为字符串内容 [15,16]。为了取得程序的控制权，需要在字符串合适的位置写入虚函数表地址，这个地址就是通过堆喷射能稳定布局的内存地址。因此传统技术中堆喷射的目的就是补全虚函数表，并布置 SHELLCODE。这类利用技术如图 2 所示。

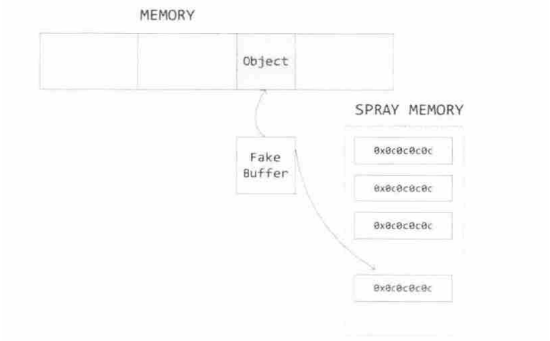

图2 堆喷射补全虚函数表

如果用某个浏览器内置对象或其他结构代替字符串去重用被释放的内存，这样就有可能不需要通过堆喷射补全虚函数表。但这样的对象或结构是有条件的 ：1）该结构的前 4 个字节是一个地址，该地址需要指向可控字符串 ；2）该结构大小可以控制，这样可以有效使被释放内存得到重用。这样的结构在 IE8 浏览器确实存在，对应的标签为 <t:ANIMATECOLOR id="myanim"/>。

该对象的 value 值是一个通过分号分隔的字符串。在对象属性 value 表示内存中，记录着每个分号分隔的字符串地址。这样 value 所占用的内存大小可以精确控制。假设 value 包含字符串个数为 N，并且一个地址为 4 字节，则占用的内存块大小 C 为 N*4。因此如果被释放的内存大小为 S，则 value 包含的字符串个数 N 为 S/4。ANIMATECOLOR 对象的 value 属性占用内存如图 3 所示。

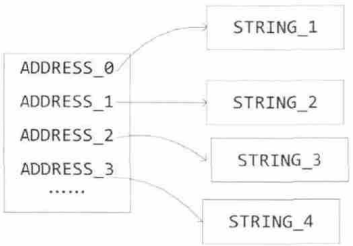

图3 value属性结构图

3 漏洞利用技术测试及实践

这种非堆喷射的 IE 漏洞利用方法的典型实践是 CVE-2012-4792 漏洞。该漏洞是一个较为典型的 UAF 漏洞，漏洞的易用性和稳定性较高。该漏洞是一个浏览器内 CButton 对象的 UAF 漏洞。通过某些 JS 操作，导致 CButton 对象被释放，而其他地方依然保留有对该内存的引用。后面的操作中，继续引用了该对象内部的数据或地址，导致漏洞产生。观察崩溃点处内存情况，发现该内存已经被释放，并且通过 CButton::vector deleting destructor 释放。CButton 对象内存操作函数调用栈如图 4 所示。

图4 CButton对象内存操作函数调用栈

对于该漏洞，通过传统的堆喷射方法可以完成漏洞利用。通过字符串重用被释放的内存一般使用 className 属性。部分代码如代码 2 所示。

代码 2

```
e_div.className = "\u0c0c\u0c0cshuishuishuishuishuishuishuishuishuishui*";
```

配合的堆喷射代码见代码 1。在调试器中观察堆喷射对该漏洞的利用效果如图 5 所示。

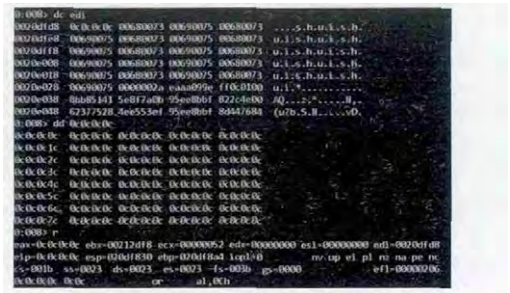

图5 堆喷射方式漏洞利用效果

由图 5 可以看出，通过堆喷射，在 0x0c0c0c0c 地址处分配了内存，并且内存数据可控。观察 EIP 也已经被劫持到 0x0c0c0c0c 地址处。可以看出堆喷射的方法基本上可以满足漏洞利用的要求。

对于非堆喷射方法，核心实现代码如代码 3 所示。

代码 3

```
animvalues = "\u4141\u4141"
        while(animvalues.length < 0xDC) {
                animvalues += animvalues
        }
        for(i = 0; i < 21; i++) {
                animvalues += ";cyan";
        }
```

可以看到 animvalues 包含了 22 个字符串，这些字符串通过分号分隔。22 个字符串可以占位 0x58 个字节，可以满足内存地址重用。animvalues 的第一个字符串比较关键，里面可以存放伪造的虚函数表和 SHELLCODE。在调试器中观察非堆喷射的漏洞利用效果如图 6 所示。

图6 非堆喷射漏洞利用效果

从图 6 可以看到，EIP 已经被劫持到 0x41414141。EDI 指向的地址为 animvalues 中各字符串指针地址。第一个地址 0x00195F88 为伪造的虚函数表，同时也可以放置 ROP 链或 SHELLCODE。由于漏洞利用过程中会调用 edi+0xDC 的函数指针，因此可以在 0x00195F88+0xDC 处放置一个指向 XCHG EAX,ESP RET 的指令地址。至此 <t:ANIMATECOLOR /> 标签基本完成了非堆喷射的漏洞利用过程。后续的绕过 DEP 及 ASLR 的技术已不在本文的讨论范围。

非堆喷射漏洞利用方法在 IE 浏览器漏洞利用中，可以做到整个利用过程瞬间完成。相比传统的堆喷射思路，其漏洞利用成功率和完成速度方面有显著的提升。

4 结束语

本文对基于堆喷射的 IE 浏览器漏洞利用技术进行了研究分析，总结了其在漏洞利用过程中的几点不足。对比堆喷射技术介绍了一种非堆喷射的 IE 浏览器漏洞利用技术。最后在 CVE-2012-4792 漏洞中得到有效测试和验证。对比堆喷射技术，该方法在执行效率和稳定性方面得到显著提升。（责编 马珂）

参考文献

[1] E. J. Schwartz, T. Avgerinos, D. Brumley. Exploit hardening Made Easy [C].Proceedings of the 20th USENIX Security Symposium, 2011.

[2] Haroon Meer. The Complete History of Memory Corruption Attacks [C]. BlackHat Confidence USA, 2010.

[3] David Litchfield, Buffer Underruns. DEP, ASLR and improving the Exploitation Prevention Mechanisms (XPMs) on the Windows platform [EB/OL].http://www.ngssoftware.com,2014-05-10

[4] Ken Johnson, Matt Miller. Exploit Mitigation Improvement in Windows 8 [C]. Blackhat USA , 2012

[5] Alexander Sotirov. Heap Feng Shui in Javascript [EB/OL]. http://www.phreedom.org/ research/heap-feng-shui/heap-feng-shui.html ,2014-03-05.

[6] Ratanaworabhan P, Livshits B , Zorn B. NOZZLE: A Defense Against Heap-spraying Code Injection Attacks [C]. Proceedings of the 18th USENIX Security Symposium, 2009.

[7] Ben Hawkes. Attacting Vista Heap [C]. Ruxcon, 2008.

[8] Adrian Marinescu Windows Vista Heap Management Enhancements [C]. Blackhat ,2006.

[9] Brett Moore. Heaps about Heaps [C]. SyScan Singapore, 2008.

[10] V Pappas, M Polychronakis. Smashing the Gadgets: Hindering Return-Oriented Programming Using In-place Code Randomization[C]. IEEE Symposium on Security and Privacy, 2012.

[11] Kangjie Lu, Dabi Zou, Weiping Wen, Debin Gao.Packed, Printable, and Polymorphic Return-Oriented Programming[C]. The 14th International Symposium on Recent Advances in Intrusion Detection, 2011.

[12] Peter Vreugdenhil. Pwn2Own-2010-Windows 7-InternetExplorer 8 [EB/OL]. http://wenku.baidu.com/link?url=7eI5HgHLc-Je2dGrPwLFUSnlaaSoJj2iqF6d9FhUgJ6IBmSpchd32zgRUFNJQ_MG5y1p_VdNTNGIf_n8LRKZRafyYJ6L6FKK3Zkuih2Mgq7,2014-05-10.

[13] Blazakis D. Interpreter exploitation: Pointer inference and JIT spraying[C]. Black Hat DC, USA, 2010.

[14] Yang Yu. DEP/ASLR Bypass without ROP/JIT [C]. CanSecWest , 2013.

[15] Chris Valasek. Understading the Low Fragmentation Heap [C]. Blackhat , 2010.

[16]Angela. Low-Fragmentation Heap [EB/OL]. http://msdn.microsoft.com/en-us brary/aa366750(v=vs.85).aspx, 2014-03-05.