

[Full-Disclosure] Listbox And Combobox Control Buffer Overflow

Source: <http://www.derkeiler.com/Mailing-Lists/Full-Disclosure/2003-10/1020.html>

From: Brett Moore (brett.moore_at_security-assessment.com)

Date: 10/16/03

To: "Full-Disclosure@Lists.Netsys.Com" <full-disclosure@lists.netsys.com>

Date: Thu, 16 Oct 2003 13:46:45 +1300

=====
= Listbox And Combobox Control Buffer Overflow
=
= MS Bulletin posted: October 15, 2003
= <http://www.microsoft.com/technet/security/bulletin/MS03-045.asp>
=
= Affected Software:
= Microsoft Windows NT 4.0
= Microsoft Windows 2000
= Microsoft Windows XP
= Microsoft Windows 2003
=
= Public disclosure on October 15, 2003
=====

As past history has shown us, Windows has many buffer overflow resulting from the mishandling of long file names or path names. This one is no different.

== Description ==

Sending either a LB_DIR message to a listbox or a CB_DIR message to a combobox, specifying a large pathname as the parameter will result in the following event log message.

Event Type: Error
Event Source: Service Control Manager
Event Category: None
Event ID: 7031
Description:
The [application] service terminated unexpectedly.

The LB_DIR and CB_DIR messages are defined as;

Full-Disclosure: [Full-Disclosure] Listbox And Combobox Control Buffer Overflow

LB_DIR

An application sends an LB_DIR message to add a list of filenames to a list box

wParam = (WPARAM) (UINT) uAttrs; // file attributes
lParam = (LPARAM) (LPCTSTR) lpszFileSpec; // filename address
lpszFileSpec
Value of lParam. Pointer to the null-terminated string that specifies the filename to add to the list

CB_DIR

An application sends a CB_DIR message to add a list of filenames to the list box of a combo box.

wParam = (WPARAM) (UINT) uAttrs; // file attributes
lParam = (LPARAM) (LPCTSTR) lpszFileSpec; // address of filename
lpszFileSpec
Value of lParam. Pointer to the null-terminated string that specifies the filename to add to the list

== Exploitation ==

On Windows 2000, the utility manager runs under the localsystem account and contains a listbox control that will accept messages from unprivileged users, allowing for the escalation of privileges to localsystem level.

The following details are based on the exploitation of that listbox.

After sending a message with a large pathname utilman will cause an exception within a call to wcsncpy.

* From MSDN *

strcpy, wcsncpy Copy a string.

```
char *strcpy( char *strDestination, const char *strSource );  
wchar_t *wcsncpy( wchar_t *strDestination, const wchar_t *strSource );
```

Parameters

strDestination – Destination string
strSource – Null-terminated source string

Remarks

The strcpy function copies strSource, including the terminating null character, to the location specified by strDestination.

No overflow checking is performed when strings are copied or appended.

* End MSDN *

The exception occurs at this code location;

```
77F81E98 mov dx,word ptr [ecx]  
77F81E9B mov word ptr [esi],dx <-- Exception  
77F81E9E inc esi
```

Full-Disclosure: [Full-Disclosure] Listbox And Combobox Control Buffer Overflow

```
77F81E9F inc esi
77F81EA0 inc ecx
77F81EA1 inc ecx
77F81EA2 test dx,dx
77F81EA5 jne 77F81E98
```

At this point ESI has been incremented to much and is now pointing to an invalid memory location. The registers look like this;

```
EAX = 007AF6DC EBX = 0000018D
ECX = 007E0924 EDX = 0000FFFF
ESI = 007B0000 EDI = 007E0000
EIP = 77F81E9B ESP = 007AF6AC
EBP = 007AFD6C EFL = 00000286
```

The area where the pathname has been copied to starts at 0x007AF6F7, which is higher than ESP, but lower than EBP. The memory starting at EBP now contains the data passed in the pathname, and any future reference to EBP will reference this data.

```
007AFD6C 58 58 58 58 58 XXXXX
007AFD71 58 58 58 58 58 XXXXX
007AFD76 58 58 58 58 58 XXXXX
007AFD7B 58 58 58 58 58 XXXXX
007AFD80 58 58 58 58 58 XXXXX
```

Because an exception has occurred, and our pathname has overwritten the exception handlers on the stack, an unhandled exception will occur when execution flow reaches;

```
77F8EB6B mov ecx,dword ptr [ebp+18h] <--- EBP points to buffer
77F8EB6E call ecx <--- We control ECX
```

At this point EBX points directly into the buffer and by correctly forming the pathname, execution flow can be directed back into our buffer.

Standard stack based overflow techniques apply and exploits can be written for either Unicode or non-Unicode depending on which API is used to send the original message.

== Solutions ==

- Install the vendor supplied patch.
- Interactive processes should not run under a higher level account.

== Credit ==

Discovered and advised to Microsoft July 14, 2003 by Brett Moore of Security-Assessment.com

Full-Disclosure: [Full-Disclosure] Listbox And Combobox Control Buffer Overflow

%-) Greetz to all the people I met in Vegas, there are to many to name.
%-) Special mention to all the boys (and girls) from eEye who showed me
%-) how to party in true Vegas style!!

== About Security-Assessment.com ==

Security-Assessment.com is a leader in intrusion testing and security code review, and leads the world with SA-ISO, online ISO17799 compliance management solution. Security-Assessment.com is committed to security research and development, and its team have previously identified a number of vulnerabilities in public and private software vendors products.

Full-Disclosure – We believe in it.

Charter: <http://lists.netsys.com/full-disclosure-charter.html>