

Full-Disclosure: [Full-Disclosure] (no subject)

[Full-Disclosure] (no subject)

Source: <http://www.derkeiler.com/Mailing-Lists/Full-Disclosure/2004-02/1340.html>

disclosure_at_ossecurity.ca

Date: 02/24/04

To: <sunglasses@bay-watch.com>, <bugtraq@securityfocus.com>

Date: Tue, 24 Feb 2004 15:01:09 -0500

We checked both EMF and WMF files out and changed around the sizes and it did not crash Windows XP (SP1, EN). From the posts on the full disclosure, it seems what you reported might be caused by other factors. Or it is exploitable on older version of XP?

Here is a list of modules loaded. XP tested (not crashing): Build 2600
xpsp1.020828-1920; SP1

92 Module: 5cb00000: C:\WINDOWS\System32\shimgvw.dll for
C:\WINDOWS\EXPLORER.EXE

93 Module: 5cb00000: C:\WINDOWS\System32\shimgvw.dll for C:\PROGRAM
FILES\INTERNET EXPLORER\IEXPLORE.EXE

Peter Huang

OSsurance, Protection Against Win32 Viruses and BOF Worms

<http://www.ossecurity.ca/>

> -----Original Message-----

> From: sunglasses@bay-watch.com [mailto:sunglasses@bay-watch.com]

> Sent: Friday, February 20, 2004 1:46 PM

> To: bugtraq@securityfocus.com

> Subject: Windows XP explorer.exe heap overflow.

>

>

>

>

> Vulnerability in XP explorer.exe image loading

> ----- Systems affected:

> Current XP – others not tested. Degree: Arbitrary code

> execution. Summary ----- A malformed .emf (Enhanced Metafile,

> a graphics format) file can cause an exploitable heap overflow in

> (or near) shimgvw.dll. Details ----- The image preview code

> that explorer uses has an exploitable buffer overflow. An .emf

> file with a "total size" field set to less than the header size

> will causes explorer.exe to crash in the heap routines – in

> classic heap overflow style that should be exploitable a la the

> RPC exploits. There are two overflows here: 1. A buffer is

[Full-Disclosure] (no subject)

Full-Disclosure: [Full-Disclosure] (no subject)

- > *allocated with the size indicated in the header (no validity*
- > *checks), then the header is copied into it – if the size is less*
- > *than the header size, that's one overflow. 2. They then proceed*
- > *to read the rest of the file*