

# Re: [Full-Disclosure] Win32 Device Drivers Communication Vulnerabilities + PoC for Symantec Norton AntiVirus '2002 (probably all versions) Device Driver

**Source:** <http://www.derkeiler.com/Mailing-Lists/Full-Disclosure/2003-08/1485.html>

---

**From:** Spiro Trikaliotis (*trik-news\_at\_gmx.de*)

**Date:** 08/21/03

To: full-disclosure@lists.netsys.com  
Date: Thu, 21 Aug 2003 22:06:46 +0200

Hello,

On Fri, Aug 02, 2002 at 10:39:44AM +0200, [SEC-LABS TEAM]: wrote:

>

> *The Sec-Labs security research group found a bug in Win32 Device Drivers Communication, the white-paper for this vulnerability can*

> *be viewed at <http://sec-labs.hack.pl>, the exploit code for Symantec Norton AntiVirus '2002 (probably all versions) Device Driver is also stored at our homepage.*

> *Enjoy.*

from the posting at the Full-Disclosure Mailing list, I found the article on <http://sec-labs.hack.pl/papers/win32ddc.php>.

In this article, you write:

"As far as I know, Windows system is not checking if the memory location (arguments/memory pointers) is valid, and it isn't a windows vulnerability? – give me a break!"

For this, I want to comment the following:

Each IRP on NT/2K/XP has a method argument, that is, the last 3 bits indicate which access method is used.

The DDK defines the following types (see devioctl.h):

```
#define METHOD_BUFFERED 0
#define METHOD_IN_DIRECT 1
#define METHOD_OUT_DIRECT 2
#define METHOD_NEITHER 3
```

The IRP you used for this proof of concept (222A87h) obviously has METHOD\_NEITHER. The DDK clearly states what these methods mean: (Windows DDK/Kernel-Mode Driver Architecture/Reference/System-Defined I/O Function Codes/IRP Function Codes and IOCTL/Defining I/O Control Codes):

"METHOD\_BUFFERED, if the driver transfers small amounts of data for the request

With this method, IRPs containing the I/O control code will supply a pointer to the buffer into which or from which to transfer data at Irp->AssociatedIrp.SystemBuffer. Most I/O control codes for device and intermediate drivers use this TransferType value.

METHOD\_IN\_DIRECT, if the underlying device driver will read a large amount of data for the request using DMA or PIO and must transfer the data quickly  
With this method, IRPs containing the I/O control code will supply a pointer to an MDL, describing the output buffer at Irp->MdlAddress.

METHOD\_OUT\_DIRECT, if the underlying device driver will write a large amount of data to the device for the request using DMA or PIO and must transfer the data quickly

With this method, IRPs containing the I/O control code will supply a pointer to an MDL, describing the data buffer, at Irp->MdlAddress.

METHOD\_NEITHER, if the driver can be sent such a request only while it is running in the context of the thread that originates the I/O control request  
Only a highest-level kernel-mode driver is guaranteed to meet this condition, so this value is seldom used for the I/O control codes passed to device drivers. With this method, the highest-level driver must determine whether to set up buffered or direct access to user data on receipt of the request, possibly must lock down the user buffer, and must wrap its access to the user buffer in a structured exception handler. Otherwise, the originating user-mode caller might change the buffered data out from under the driver or the caller could be swapped out just as the driver is accessing the user buffer."

There are even more explicit texts.

In fact, METHOD\_NEITHER means literally what the name means: Windows does not touch or check the buffer pointer or the buffer itself in any way, does not perform any checks nor anything else. All text books I've read so far even advice not to use METHOD\_NEITHER unless one knows exactly what he is doing – something the driver writer obviously did not in your example.

The DDK even contains the following statement  
(Windows DDK/Kernel-Mode Driver Architecture/Design-Guide/Driver Programming Techniques/Driver Reliability Issues/Errors in Referencing User-Space Addresses):

Errors in Referencing User-Space Addresses

A driver should validate any address in user space before trying to use it.

The I/O Manager does not validate such addresses, nor does it validate

pointers that are embedded in buffers passed to drivers.

Failure to Validate Addresses Passed in Type3 (METHOD\_NEITHER) IOCTLs and FSCTLs

The I/O Manager does no validation whatsoever for METHOD\_NEITHER IOCTLs and FSCTLs. To ensure that user-space addresses are valid, the driver must use the ProbeForRead and ProbeForWrite routines, enclosing all buffer references in try/except blocks.

So, unless you find another IOCTL with another method, I think it is not correct to call this a windows vulnerability. The driver writer \*explicitly\* chose to shut down any tests Windows makes, but did not handle this case correctly. You can blame Microsoft for very much, but how can you blame Microsoft for this?

Curious,  
Spiro.

---

Full-Disclosure – We believe in it.

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