

EEYE: Microsoft ASN.1 Library Length Overflow Heap Corruption

Source: <http://www.derkeiler.com/Mailing-Lists/NT-Bugtraq/2004-02/0010.html>

From: Marc Maiffret (*mmaiffret_at_EEYE.COM*)

Date: 02/10/04

Date: Tue, 10 Feb 2004 10:22:26 -0800

To: NTBUGTRAQ@LISTSERV.NTBUGTRAQ.COM

Microsoft ASN.1 Library Length Overflow Heap Corruption

Release Date:

February 10, 2004

Date Reported:

July 25, 2003

Severity:

High (Remote Code Execution)

Systems Affected:

Microsoft Windows NT 4.0 (all versions)

Microsoft Windows 2000 (SP3 and earlier)

Microsoft Windows XP (all versions)

Software Affected:

Microsoft Internet Explorer

Microsoft Outlook

Microsoft Outlook Express

Third-party applications that use certificates

Services Affected:

Kerberos (UDP/88)

Microsoft IIS using SSL

NTLMv2 authentication (TCP/135, 139, 445)

Preamble:

We wanted to do another Night Before Xmas but the vendor missed the last few release dates, so we had to resort to our MC(SE).

U Can't Trust This

By: MCSE Hammer

NT–Bugtraq: EEYE: Microsoft ASN.1 Library Length Overflow Heap Corruption

Blaster did ya some harm
We just say, hey, another worm
But thank you, for trusting me
To mind your site's security
It's all good, when your server's downed
Our dope PR will pass blame around
Cuz it's known as such
That this is some software, you can't trust

I told ya Homeland
U can't trust this
Yeah that's why we're giving ya the code
U can't trust this
Check out eEye, man
U can't trust this
Yo let 'em bust more funky system
U can't trust this

Give 'em a string or rcvfrom
Like no sweat they got the keys to your kingdom
Now ya know
You talk about eEye, you're talking about holes
Remote and tight
Coders still sweating so someone better write
A book to learn
What it's gonna take in '04
To earn some trust
Legit, either secure or ya might as well quit

That's the word because you know
U can't trust this
U can't trust this

Breakin' in

Stop — eEye time

Description:

eEye Digital Security has discovered a critical vulnerability in Microsoft's ASN.1 library (MSASN1.DLL) that would allow an attacker to overwrite heap memory on a susceptible machine and cause the execution of arbitrary code. Because this library is widely used by Windows security subsystems, the vulnerability is exposed through an array of avenues, including Kerberos, NTLMv2 authentication, and applications that make use of certificates (SSL, digitally–signed e–mail, signed ActiveX controls, etc.).

Technical Description:

The MSASN1 library is fraught with integer overflows. In this advisory, we'll describe a pair of arithmetic errors in a generic and low–level part of ASN.1 BER decoding that allow a very large swath of heap memory

to be overwritten. This vulnerability affects basically any client of MSASN1.DLL, the most interesting of which are LSASS.EXE and CRYPT32.DLL (and therefore any application that uses CRYPT32.DLL).

Although the specifics of ASN.1 BER encoding are beyond the scope of this advisory, the basic idea is that it's an encoding scheme for flexibly representing binary data, and is often compared to "binary XML." Each piece of data is encoded as a typed value, which is constructed as a tag number that describes how to interpret the following value data, then the length of the data, and finally, the data itself. This length field is the subject of our advisory. By supplying a very large value (from 0xFFFFFFFFD to 0xFFFFFFFF) in this field, we can cause an integer overflow in a heap allocation routine, and although there are checks in place to ensure the validity of a value's length, a separate pointer arithmetic overflow in the verification routine gives rise to the vulnerability. Here's how:

1. When a simple value (a value that consists of atomic data, rather than more values) is decoded by MSASN1, ASN1BERDecLength() is called to retrieve the length of the value, then passes the reported length to the ASN1BERDecCheck() function to make sure that that much data actually exists.

2. ASN1BERDecCheck() verifies that (pointer_to_start_of_data + reported_length_of_data), unsigned, is less than or equal to (pointer_to_start_of_BER_block + total_size_of_BER_block). If not, the function returns failure, which propagates back up the call stack and causes decoding to stop. (As an aside, it's interesting to note that this vulnerability was silently fixed in Windows 2000 SP4 and Windows Server 2003, due to an additional comparison being included in ASN1BERDecCheck().)

3. If the function that called ASN1BERDecLength() then attempts to allocate memory and make a copy of the data (e.g., ASN1BERDecOctetString(), but not the ASN1BERDecOctetString2() variant), it will then pass the decoded length to DecMemAlloc(), which rounds the length up to a DWORD multiple and then attempts to allocate the result. The operation of this function can be represented as "LocalAlloc(LMEM_ZEROINIT, (length + 3) & ~3)."

4. If DecMemAlloc() succeeds, the calling function then memcpy()'s the value data into the allocated heap buffer, using the original decoded length of the value as the byte count.

If a very large length is decoded by ASN1BERDecLength() in step 1, then there will be an integer overflow when ASN1BERDecCheck() adds the length to the current data pointer in step 2, essentially causing the resulting pointer to "wrap around" the 32-bit address space and therefore have an address that is numerically less than the pointer to the end of the buffer.

NT–Bugtraq: EEYE: Microsoft ASN.1 Library Length Overflow Heap Corruption

Now, to be more specific, if a length in the range 0xFFFFFFFF through 0xFFFFFFFF is given, it will pass through ASN1BERDecCheck() with no problem, and then something really bad happens. Because of the round–off in DecMemAlloc(), the three lengths in this range will all round "up" to zero. LocalAlloc() successfully allocates a zero–length heap block whose address gets returned to the caller, but then the original, very large length is handed to memcpy(). The result is a classic, complete heap overwrite, where all contiguous heap memory following the zero–length block is wiped out by arbitrary data.

The simplest way to manifest this condition is to encode a simple octet string (tag 04h) with a length–of–length set to 4 and a length of 0xFFFFFFFF, which corresponds to the bytes 04h/84h/FFh/FFh/FFh/FFh. Depending on which decoder functions the MSASN1 client uses, it is also possible to leverage this vulnerability through OIDs and character strings as well. The following is a sampling of vulnerable decoder functions:

- ASN1BerDecCharString
- ASN1BERDecChar16String
- ASN1BERDecChar32String
- ASN1BERDecEoid
- ASN1BERDecGeneralizedTime
- ASN1BERDecMultibyteString
- ASN1BERDecOctetString
- ASN1BERDecOpenType
- ASN1BERDecSXVal
- ASN1BERDecUTCTime
- ASN1BERDecUTF8String
- ASN1BERDecZeroCharString
- ASN1BERDecZeroChar16String
- ASN1BERDecZeroChar32String
- ASN1BERDecZeroMultibyteString

Note: Due to the technical nature of the vulnerability described above, this advisory may contain disassembly and/or hexadecimal byte codes. This information is in no way related to "exploit code", "payloads", or "shell code".

Protection:

Retina Network Security Scanner has been updated to identify this vulnerability:

<http://www.eeye.com/html/Products/Retina/index.html>

Vendor Status:

Microsoft has released a patch for this vulnerability. The patch is available at:

<http://www.microsoft.com/technet/security/bulletin/MS04–007.asp>

Credit:

Discovery: Derek Soeder

NT-Bugtraq: EEYE: Microsoft ASN.1 Library Length Overflow Heap Corruption

Additional Research: Yuji Ukai

Very Special Thanks:

Yuji Ukai, again, for the majority of "The Menu"

Steve Peters and Shawn O'Donnell for extreme ASN.1 (BER!) and certificate lore

Greetings:

DDDDDD; GM-TX, BMNN-FL, JKP-FL, NV-TX; Heather & Heather; all the coffee shop folks; 214; and always, every last one of the eEye crew

Copyright (c) 1998-2004 eEye Digital Security

Permission is hereby granted for the redistribution of this alert electronically. It is not to be edited in any way without express consent of eEye. If you wish to reprint the whole or any part of this alert in any other medium excluding electronic medium, please e-mail alert@eEye.com for permission.

Disclaimer

The information within this paper may change without notice. Use of this information constitutes acceptance for use in an AS IS condition. There are NO warranties with regard to this information. In no event shall the author be liable for any damages whatsoever arising out of or in connection with the use or spread of this information. Any use of this information is at the user's own risk.

Feedback

Please send suggestions, updates, and comments to:

eEye Digital Security

<http://www.eEye.com>

info@eEye.com

NTBugtraq Editor's Note:

Most viruses these days use spoofed email addresses. As such, using an Anti-Virus product which automatically notifies the perceived sender of a message it believes is infected may well cause more harm than good. Someone who did not actually send you a virus may receive the notification and scramble their support staff to find an infection which never existed in the first place. Suggest such notifications be disabled by whomever is responsible for your AV, or at least that the idea is considered.
