binamuse
security research

IBM Lotus Notes

PNG Integer Overflow

May 2013

IBM Lotus Notes is the client of a collaborative client-server plataform, being IBM Lotus Domino the application server. The email-client capability is one of its most important and used features. IBM Lotus Notes fails to correctly parse a PNG image file embedded in an email. Arbitrary code execution is proved possible after a malicious email is opened or just previewed.

Juan Pablo De Francesco
<jpdefrancesco@binamuse.com>

1 Description

Title	PNG Integer Overflow
Product	IBM Lotus Notes
Version	8.5.3 v.20110916.0921 (FP1/FP2/FP3 also vulnerable)
Homepage	http://www.ibm.com/software/lotus/products/notes/
Binary affected	nnotes.dll
Binary MD5	[CDC711D74092BB89FEF4D78C6927B119]

Title	PNG Integer Overflow
Product	IBM Notes
Version	9.0 (20121208.0805)
Homepage	http://www.lotus.com/ldd/ndsebetaforum.nsf/
Binary affected	nnotes.dll
Binary MD5	[EBC6FDDDD06EDF637803FE683C5B2F96]

2 Vulnerability brief information

Vulnerability Class	Remote Heap Overflow
Affected Versions Tested	8.5.3 | 9.0
Affected Platforms	Windows
	Linux
	OSX
Reliability Rating	100%
Configuration Requirements	None
Attack Vector	Mail/Network
Exploitation Impact	Code execution
Patch	`http://www-01.ibm.com/support/docview.wss?uid=swg21636024`
CVE	CVE-2013-2977
Reference	`http://blog.binamuse.com/2013/05/lotus-notes-cve-2013-2977.html`

3 Common Vulnerability Scoring System

Base Metrics		
Access **V**ector	**N**etwork	The vulnerability is exploitable with network access
Access **C**omplexity	**L**ow	Specialized access conditions or extenuating circumstances do not exist
Authentication	**N**one	Authentication is not required to exploit the vulnerability
Confidentiality Impact	**C**omplete	There is total information disclosure, resulting in all system files being revealed
Integrity Impact	**C**omplete	There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised. The attacker is able to modify any files on the target system
Availability Impact	**C**omplete	There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable

Temporal Metrics		
Exploitability	**H**igh	Fully functional exploit and details are widely available
Remediation **L**evel	**U**navailable	There is either no solution available or it is impossible to apply
Report **C**onfidence	**C**onfirmed	The vulnerability has been acknowledged by the vendor or author of the affected technology

Environmental Metrics		
Collateral **D**amage **P**otential	**M**edium-**H**igh	A successful exploit of this vulnerability may result in significant loss of revenue or productivity
Target **D**istribution	**H**igh	Between 76% - 100% of the total environment is considered at risk

4 Vulnerability Workaround

No workarounds for this vulnerability are known.

5 Vulnerability Details

IBM Lotus Notes has an email client that allows embedded images in the email's body. One of the images format supported is PNG, and its parsing is managed by `libpng`. The problem exposed here isn't within `libpng`, but in the way that IBM Lotus Notes use the interface exposed by `libpng`.

The faulty routine is located at `nnotes+0x607B60` and looks like:

```
...
mov     eax, [ebx+0Ch]
push    edi
push    eax
push    ecx
call    png_get_rowbytes
mov     edx, [ebx+0Ch]
mov     edi, eax                ;edi = width*4
mov     eax, [ebx+8]
push    edx
push    eax
mov     [ebp+var_28], edi
call    png_get_channels
movzx   cx, al
mov     eax, [esi+4]            ;eax = height
imul    eax, edi                ;eax = (width*4)*height
add     esp, 10h
lea     edx, [ebp+var_14]
push    edx
push    eax
push    eax
push    10000h
mov     [esi+17h], cx
call    OSMemAllocRaw           ;alloc (width*4)*height bytes
...
```

An integer overflow may occurs in the multiplication `(width*4)*height`. In fact, the dimensions constraints are:

$$0 < width \leq 0\text{xF4240}$$
$$0 < height \leq 0\text{x1FFFFF7E}$$

Therefore, the allocated size within `OSMemAllocRaw` is *almost* completely controlled. That buffer will be used to store all decompressed rows of the image. And will be filled by `libpng` in reverse order, i.e. writing the last `width*4` bytes and continuing upwards `height` times.

As we can see, with certain values of `with` and `height` we can have a heap *backward* overflow, in this scenario we control:

- Overflowed buffer's size (*with restrictions*)
- Overflow size (*with restrictions*)
- Overflow data (*completely*)

A similar situation occurs in IBM Notes.

6 Exploitation

To trigger the parsing of an image attached to an email, we use the fact that IBM Lotus Notes parses `HTML` content. We achieve that sending an `img` tag with its `src` referencing the attached file.

As we said before, the misallocated buffer is used to store all the image's rows. There exists a second call to `OSMemAllocRaw` that will allocate a buffer of `4*height` bytes, i.e. a `DWORD` for each row, that `DWORD` will store a pointer to begining of each row in the first allocated buffer.

If we can sort the memory in such a way that the misallocated buffer is after the pointers buffer, we could possibly turn this bug to an arbitrary write. In other words, we could overwrite

a row's pointer with an arbitrary address, then `libpng` eventually will use that address to copy the corresponding decompressed row, overwriting `4*width` bytes starting at that address.

To achieve the stated before, we need a way to sort the memory as we want. We build a way playing with `JavaScript` and reversing its garbage collector, to trigger it when we want.

Now we have to find what we could overwrite to be as stealth as possible, and to get something reliable in both IBM Lotus Notes and IBM Notes. They share the following modules (no ASLR, no Rebase):

`LTOUIN22.dll` base at `0x62990000`
`MSVCR71.dll` base at `0x7C340000`

We decide to overwrite the function pointer located at `0x629B9184`. That function will be called eventually and then the ROP chain begins.

It's important to note that additional to the actual payload, some code is executed to assure the maximum stealthiness and to restore the state to thread and module involved.

A Proof of Concept exploit is provided and tested for Windows 7 and Windows XP.

This was tested opening and/or previewing the email.

If everything went ok the PoC will run a calculator. It was tested on fresh installations of Windows 7 Ultimate SP1 both 32 a 64 bit and Windows XP Pro SP3 32 bit versions.