

[NT] PNG (Portable Network Graphics) Deflate Heap Corruption Vulnerability (Windows)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-12/0035.html>

From: support@securiteam.com

Date: 12/12/02

From: support@securiteam.com

To: list@securiteam.com

Date: 12 Dec 2002 10:45:31 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

Beyond Security would like to welcome Tiscali World Online to our service provider team.

For more info on their service offering IP-Secure, please visit http://www.worldonline.co.za/services/work_ip.asp

PNG (Portable Network Graphics) Deflate Heap Corruption Vulnerability (Windows)

SUMMARY

During a review of the PNG image format implemented in Microsoft Windows, two separate vulnerabilities were discovered related to the interpretation of PNG image data. The first vulnerability deals with the handling of the IDAT header and does not appear to be of significant threat level. The second vulnerability can be exploited to execute code when the malicious PNG image is viewed. Due to the complexity of each of these vulnerabilities eEye decided only to describe the latter in detail.

DETAILS

Vulnerable systems:

- * Microsoft Internet Explorer 5.01
- * Microsoft Internet Explorer 5.5
- * Microsoft Internet Explorer 6.0

General Description:

A heap corruption vulnerability exists due to the way the function `inflate_fast()`, within `pngfilt.dll`, handles certain invalid data present in "deflate" data input streams in a PNG image file. The "deflate"

Securiteam: [NT] PNG (Portable Network Graphics) Deflate Heap Corruption Vulnerability (Windows)

compression specification allows for the repetition of patterns that occur in the decompressed data. This is accomplished by specifying a pair of special codes that tell the decompression routine how far back into the decompressed stream the pattern occurred (distance code), and the length of the pattern to repeat in bytes (length code). The `inflate_fast()` routine does not properly handle length codes marked in the specification as invalid, and as a result, a pattern can be replicated over a large portion of the heap, allowing a skilled attacker to redirect the execution of a thread into a "deflated" payload embedded in the deflate datastream within the malicious PNG image.

Technical Description:

The heap overflow described above occurs in the interpretation of a compressed block that uses fixed Huffman codes (`BTYPE = 1`). Length codes #286 and #287, while labeled as invalid in the formal specification (RFC 1951), are not discarded by the inflation routine, and are instead treated as zero-length codes. However, due to the way the inflation routine is designed (see below), the length counter is decremented prior to being evaluated, and an integer overflow will occur. As a result, the loop will attempt to repeat the pattern we specify over all 4GB (`0xFFFFFFFF`) of virtual address space, filling our 32KB output buffer and proceeding to overwrite process memory until finally reaching an invalid page in memory and producing a fault.

The problem code is presented below in assembly, and C pseudo code.

Pattern-repetition loop in `PNGFILT.DLL`. version 5.0.2920.0:

```
69198FAF mov ecx,dword ptr [ebp+8]
69198FB2 mov cl,byte ptr [ecx]
69198FB4 mov byte ptr [edi],cl
69198FB6 inc edi
69198FB7 inc dword ptr [ebp+8]
69198FBA dec dword ptr [ebp+0Ch]
69198FBD jne 69198FAF
```

Pseudo-code representation of previous assembly:

```
do
{
    *dest = *src;
    ++dest;
    ++src;
}
while (--len);
```

After the process heap following the 32kb output buffer has been overwritten, numerous threads running within the Internet Explorer process attempt to free heap blocks whose memory management structures have been overwritten. By supplying a carefully crafted memory management header, we can alter any 32-bit address to which we have write access in Internet Explorer's virtual address space.

Securiteam: [NT] PNG (Portable Network Graphics) Deflate Heap Corruption Vulnerability (Windows)

For the sake of demonstration, we will hijack the hook for the unhandled exception filter, overwriting the pointer to the handler with the address of a "CALL DWORD PTR [ESI+0x4C]" instruction present in MSHTML.DLL. We explain the purpose of this particular instruction below.

One of the threads that attempts to free a heap block with a memory management structure we now control, will cause the unhandled exception filter hook to be overwritten and will then cause an exception to be thrown by accessing an invalid address. This exception is thrown due to the following code sequence within RtlAllocateHeap() in NTDLL.DLL:

```
77FCB3F5 mov [ecx], eax
77FCB3F7 mov [eax+4], ecx
```

During this operation the eax register is set to the address within MSHTML.DLL where our "call dword ptr [esi+0x4c]" resides. The ecx register is set to the address of the unhandled exception filter hook.

After the first operation overwrites our exception filter hook, the second operation will generate an exception when it attempts to "mov" our exception handler address four bytes after the address we specified in the ".text" section of MSHTML.DLL. An exception is generated due to the fact that code sections, or ".text" sections are loaded into a process with read-only permissions.

The exception created by this operation is unhandled. The faulting thread will then attempt to call the unhandled exception filter. Since the address of this function has been changed, execution will redirect to the "call dword ptr [esi+0x4c]".

When an unhandled exception occurs, the unhandled exception filter is executed and receives, as an argument, an exception record. The "call dword ptr [esi+0x4c]" will redirect execution to an address supplied within the exception record. The data at this address is part of our decompressed stream.

The repeated pattern in our decompressed stream contains the two addresses used to overwrite the unhandled exception filter hook, along with a padding instructions and an unconditional "jmp" that will direct execution up what is essentially a jump chain formed by the pattern repetition, into the beginning of our deflated datastream and the deflated payload of choice.

During tests in our lab we noticed that under certain circumstances, race conditions occur that make exploitation very difficult. We developed intermediate solutions to these by reconstructing objects in heap so that the conflicting threads would continue long enough for our target thread to be exploited.

Mitigating Factors:

It should be noted that due to memory management system behavior across

Securiteam: [NT] PNG (Portable Network Graphics) Deflate Heap Corruption Vulnerability (Windows)

various Windows operating system environments, exploitation may become extremely difficult and in some cases unreliable.

Vendor Status:

Microsoft was contacted in August 2002. Internet Explorer Service Pack 1 eliminates this vulnerability. Internet Explorer Service Pack 1 can be retrieved using the following URL:

<<http://microsoft.com/windows/ie/downloads/critical/ie6sp1/default.asp>>
<http://microsoft.com/windows/ie/downloads/critical/ie6sp1/default.asp>

Microsoft has released a security bulletin for this flaw. It is located here:

<<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-066.asp>>
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-066.asp>

ADDITIONAL INFORMATION

The information has been provided by <<mailto:marc@eeye.com>> Marc Maiffret of eEye.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.