

# [NT] Technical Description of the SSL PCT Vulnerability

**Source:** <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-05/0019.html>

---

**From:** SecuriTeam ([support\\_at\\_securiteam.com](mailto:support_at_securiteam.com))

**Date:** 05/04/04

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: 4 May 2004 15:57:47 +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>

-----

Technical Description of the SSL PCT Vulnerability

---

## SUMMARY

A remotely exploitable buffer overflow condition was found in Microsoft's IIS and reported in a previous <http://www.securiteam.com/windowsntfocus/5CP0L0KCKO.html> article. A more thorough and detailed analysis of the vulnerability in MS's SSL library is presented below.

## DETAILS

Vulnerable Systems:

- \* Microsoft Windows 2000 up to and including SP4
- \* Microsoft Windows NT version 4 up to and including SP6a
- \* Microsoft Windows XP up to SP1
- \* Microsoft IIS versions 4.1, 5.0 and 5.1
- \* Microsoft Exchange Server version 5.0 with SSL enabled
- \* Active Directory with SSL enabled

Note: The SSL library included in Windows Server 2003 contains the vulnerability. However, the PCT 1.0 protocol is disabled by default. In order to perform successful exploitation, the PCT protocol must be enabled

and a valid certificate must be installed.

Analysis:

The vulnerable code is located in schannel.dll that is loaded by LSASS.exe, a disassembly of the vulnerable code translates roughly to the following C code:

```
function(char *packet, unsigned int N)
    char buf[32];
    unsigned int register i;
    if(N < 32)
    {
        memcpy(buf,packet,N);
        for(i = 0; i < N; i++)
            buf[i+N] = ~buf[i];
    }
```

Compiling the above code with optimizations and assembly inlining should give the following assembly code, equivalent to the code found in LSASS.exe:

```
text:781786C8 mov [ebp-60], eax
text:781786CB mov eax, [esi+0Ch]
text:781786CE mov ecx, eax
text:781786D0 add esi, 30h
text:781786D3 mov edx, ecx
text:781786D5 lea edi, [ebp-24]
text:781786D8 shr ecx, 2
text:781786DB rep movsd
text:781786DD mov ecx, edx
text:781786DF and ecx, 3
text:781786E2 rep movsb
text:781786E4 xor esi, esi
text:781786E6 test eax, eax
text:781786E8 jbe short dontcopy
text:781786EA
text:781786EA loop:
text:781786EA mov dl, [ebp+esi-24]
text:781786EE lea ecx, [ebp+esi-24]
text:781786F2 inc esi
text:781786F3 not dl
text:781786F5 cmp esi, eax
text:781786F7 mov [ecx+eax], dl
text:781786FA jb short loop
text:781786FA dontcopy:
```

In the above C code, the variable N is taken from the packet itself. A value of N larger than 0x10 is enough to trigger an overflow, while a value of 0x16 is enough to overwrite the return address of the function. The boundary check (N<32) is done before the call to memcpy() but the concatenation code that follows it is logically wrong. The vulnerability could be exploited to execute arbitrary code. For that purpose the return address should be overwritten with an address pointing to data controlled

## Securiteam: [NT] Technical Description of the SSL PCT Vulnerability

by the attacker, but the address of such data is not predictable.

One way to do this on Windows is to redirect the program flow (by overwriting of the function return address) to a known address in order to redirect the code back to the attacker's code. The known address belongs, by definition, to the vulnerable process or to a module loaded by it. The code location can change for various applications and system levels so care must be taken when constructing an appropriate payload.

In this case, when the vulnerable function returns to the address chosen by the attacker, there is no register pointing to the controllable data, but there is a pointer to the PCT packet on the thread's stack at [esp+6c]. If we can find a set of instructions in the vulnerable application memory equivalent to CALL [esp+6C] part of the job is done. One can use the following instructions as a pattern (which LSASS.exe contains in numerous locations):

```
add esp,6c
ret
```

But there is still a problem to solve: The address at [esp+6c] points to the PCT packet header, therefore the fields of that header are going to be used as executable code. If the PCT packet doesn't fulfill some protocol checks, execution flow will not reach the vulnerable function. We need to craft a packet with valid field values and valid opcodes.

In this context a "valid field value" is one that allows the execution flow to reach the vulnerable function AND is also a valid opcode so as to prevent the application from crashing when it is executed. Note that for successful exploitation, it's not necessary to craft a packet that complies with the PCT RFC. The packet data used by the first exploit made is:

```
"\x80\x66\x01\x02\xbd\x00\x01\x00\x01\x00\x16\x8f\x86\x01\x00\x00\x00"
```

Value Condition to fulfill  
XX YY (RecordLength):

```
if(XX & 0x80)
    RecordLength = ((XX & 0x7f) << 8) | YY
else
    RecordLength = ((XX & 0x3f) << 8) | YY
```

Selecting XX=0x80 and YY=0x66 will satisfy the following conditions:

01 0x01 (j):= 0x01

02 BD 0x02bd (k):0x0002 <= k < 0x301 < 0x8001

Here two bytes that satisfy the above condition can be used, 0x02 and 0xbd are just arbitrarily chosen values. Note that 0xbd == mov ebp.

00 01 0x0001 (l): > 0x0001, its better to keep it under 0x0003

00 01 0x0001 (m):<= 0x10

00 16 0x001A (N): (1+m+N+9 <= RecordLength) && (0 < N <= 0x20)

8F 0x008F (o): = 0x008F

## Securiteam: [NT] Technical Description of the SSL PCT Vulnerability

86 01 0x8601 (p): >= 0x8001

Additionally, RecordLength must be less or equal to the packet size. The packet then translates to the following assembly code:

```
80660102 and byte ptr [esi+0x1],0x2
bd00010001 mov ebp,0x1000100
0016 add [esi],dl
8f8601000000 pop [esi+0x1]
eb20 jmp 0016f40b
```

The ESI register points to writeable data in both XP and 2000, the register used by the first and fourth instructions can be changed but the third (00 16) is the right size to overwrite the return address. A jump opcode (eb xx) could be used in the first two bytes if the packet's length is modified accordingly.

Looking at the conditions that must be met for each field we can see that there are more than 25 millions different packets that will trigger the vulnerability (combinations of the possible values for XX, YY, k, l, m and p). Modifying the value for N is also possible but would require additional payload to overwrite portions of the memory of the running process with valid data.

### Detection of Possible Attack

In view of the above, detection of an attack that exploits this vulnerability should not rely entirely on packet bytes that can have value arbitrary chosen. A proper check should \*at least\* check for the required fixed values:

```
o == 0x8F
0x10 < N <= 0x20 (a value less than 0x10 does not overwrite the stack)
```

Relying on other packet bytes for a proper detection signature should be subject to careful analysis as there might be other execution paths reaching the vulnerable function.

### References

<<http://www.microsoft.com/technet/security/bulletin/MS04-011.msp>>  
<http://www.microsoft.com/technet/security/bulletin/MS04-011.msp>

<<http://www.develop.com/books/pws/draft-benaloh-pct-01.txt>>  
<http://www.develop.com/books/pws/draft-benaloh-pct-01.txt>

<<http://www.graphcomp.com/info/specs/ms/pct.htm>>  
<http://www.graphcomp.com/info/specs/ms/pct.htm>

<<http://www.coresecurity.com/products/coreimpact/index.php>>  
<http://www.coresecurity.com/products/coreimpact/index.php>

### ADDITIONAL INFORMATION

Securiteam: [NT] Technical Description of the SSL PCT Vulnerability

The information has been provided by <mailto:juliano.rizzo@corest.com>  
Juliano Rizzo, (Core Security Technologies).

=====

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.