COSEINC WINDOWS ADVISORY #2

SChannel Off-By-One Heap Corruption

Discovery Date:
28th August 2006

Date reported to Microsoft:
19th March 2007

Summary:
The Secure Channel (SChannel) library on WinXP-SP1/SP2 is vulnerable to a off-by-one heap buffer overwrite. The SChannel library implements PCT/TLS/SSL protocols exported via the Security Service Provider Interface (SSPI). It is one of several Security Service Providers loaded-in and supported by the privileged Local Security Authority (Lsass.exe) process.

In SChannel's implementation of the client-side SSLv3 handshake protocol, specifically in the processing of the server-key-exchange SSL handshake record, there is insufficient checks for malformed server-sent digital signature, with its length-field set to 0. This results in a allocation of a 0-length heap buffer (with a valid heap address). A reverse memory copy is then performed to copy-in the digital signature, by decrementing the 0-length by 1. This results in an integer-underflow, causing the heap-buffer pointer to decrement before its start address, ultimately leading to an overwrite of exactly one-byte of user-controlled value, into the heap control-block. Depending on the robustness of the application in question, this may lead to an unrecoverable heap corruption condition, causing the application to terminate. In the case of Lsass.exe on WinXP-SP2, we can crash it locally after several iterations, from a less-privileged user, causing a system reboot. Vulnerable code although also exists in WinXP-SP1 but it does not cause an unrecoverable heap corruption in Lsass.exe.

Vendor Affected:
Microsoft

Systems Affected:
WinXP-SP2 (DOS/Reboot)
WinXP-SP1 (minimal impact)

Exploitation:
1) For local machine reboot via normal user account, on WinXP-SP2
OR
For remote machine reboot by enticing user to visit HTTPS site via IE, on WinXP-SP2 (but over 2000 iterations required)

POC (crash-test/reboot):
1) Run sctest.exe from a normal user account, on client machine running WinXP-SP2.

2) sctest.exe will attempt to use SChannel's SSL implementation to parse pre-generated malformed SSL handshake records, over several iterations, causing multiple off-by-one overwrites with 0xFF byte, within the Lsass.exe process.

3) Attach Debugger to Lsass.exe to see crash. The system will notify the user and perform a 60sec. reboot count-down, after detecting the Lsass.exe crash.

** Lsass.exe crash-test can also be done by forcing/enticing Internet Explorer to access a HTTPS site, serving out the same malformed SSL handshake records (as shown in source code below). However, over 2000 iterations are needed (IE needs to access HTTPS site over 2000 times), before Lsass.exe heap corruption occurs.

Vulnerability Analysis:
(Based on schannel.dll/v5.1.2600.2180/WinXP-SP2)
The vulnerability exists in schannel.dll component that implements the SSPI-compliant PCT/TLS/SSL protocol handling implementation. For more information on SSPI and how it relates to LSA, refer to
1) http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secauthn/security/authentication_packages.asp
2) http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secauthn/security/sspi.asp

Essentially, in the case of SSPI authentication libraries like schannel, kerberos, msv1_0 (ntlm), data is exchanged between less-privileged user applications requring authentication, and Lsass.exe. With LSA providing the authentication back-end support. Both LSA and the less-privileged application communicate indirectly via the SSPI interface. Specifically, in SSL authentication, untrusted SSL record packets are passed from the less-privileged application to the privileged LSA. While extensive efforts are made in LSA to validate the SSL records, on WinXP's version of schannel, an off-by-one vulnerability exists in the parsing of the less-common and less-used SSL server-key-exchange record.
The vulnerability can hence be triggered via less-privileged client applications utilizing the schannel's client-side SSL protocol implementation. This includes Internet Explorer, whenever the user uses IE to browse a HTTPS site.

The vulnerable code exists in the _ReverseMemCopy() function and is reachable from Ssl3ParseServerKeyExchange():
(via SPProcessHandshake()->PkcsGenerateClientExchangeValue())

```
; On WinXP-SP1, the code below is located at 0x767FF976 (no symbols available)
Ssl3ParseServerKeyExchange()
...
.text:767FFFC8              movzx   ebx, byte ptr [esi]      ; MSB-byte of malformed signature length field
.text:767FFFCB              movzx   eax, byte ptr [esi+1]    ; LSB-byte of malformed signature length field
.text:767FFFCF              shl     ebx, 8
.text:767FFFD2              add     ebx, eax
.text:767FFFD4              push    ebx                      ; size=0
.text:767FFFD5              call    _SPExternalAlloc@4       ; HeapAlloc will return a valid 0-length heap buffer address
.text:767FFFDA              test    eax, eax
.text:767FFFDC              mov     [ebp+pbSignature], eax
.text:767FFFDF              jz      loc_768000B9
.text:767FFFE5              push    ebx                      ; size=0
.text:767FFFE6              lea     ecx, [esi+2]             ; address of the signature data in our malformed record
                                                             ; containing 0xFF,0x41,0x41...
.text:767FFFE9              push    ecx
.text:767FFFEA              push    eax                      ; 0-length heap buffer
.text:767FFFEB              call    _ReverseMemCopy@12
      _ReverseMemCopy()
      .text:767FF46F              mov     edi, edi
      .text:767FF471              push    ebp
```

```
.text:767FF472          mov     ebp, esp
.text:767FF474          mov     eax, [ebp+arg_8]
.text:767FF477          mov     ecx, [ebp+arg_4]
.text:767FF47A          push    esi
.text:767FF47B          mov     esi, [ebp+arg_0]
.text:767FF47E          lea     eax, [esi+eax-1]        ; EAX=0, ESI which points to 0-length
heap buffer
                                                        ; is decremented to, before start of heap buffer
.text:767FF482          mov     dl, [ecx]
.text:767FF484          mov     [eax], dl               ; Off-by-one overwrite with 0xFF from our
signature data
.text:767FF486          dec     eax
.text:767FF487          inc     ecx
.text:767FF488          cmp     eax, esi
.text:767FF48A          jnb     short loc_767FF482      ; Just one-byte overwrite!
.text:767FF48C          pop     esi
.text:767FF48D          pop     ebp
.text:767FF48E          retn    0Ch
```

~end

Credit:
This vulnerability was discovered by steven , a Windows security researcher of the COSEINC Vulnerability Research Lab (VRL).