Windows network services internals

Jean-Baptiste Marchand

Revision History
22 October 2003
Initial version.
5 July 2004
Major update
19 October 2004
Port to the docbook typesetting system
20 January 2005
Major update of the NULL sessions section, including new information about Windows XP SP2 and Windows Server 2003 SP1
19 March 2005
Additional details about NULL session restrictions for samr and lsarpc interfaces on Windows XP and Windows Server 2003 (including for Active Directory domain controllers)
31 May 2005
Many small fixes, additions and reordering of sections
May 2006
Major update: new MSRPC interfaces, Windows Vista content (SMB 2.0, new MSRPC interfaces), documentation of Windows API, MSRPC vulnerabilities section, MSRPC and DCOM network traffic sections, sections reorganization, new naming convention for generated HTML pages, URL updates.
$LastChangedDate: 2006-05-22 13:21:48 +0200 (Mon, 22 May 2006) $
Last update

Table of Contents

1. Introduction
2. TCP/IP stacks
    - 2.1. Introduction to Windows TCP/IP stacks
    - 2.2. Windows 2000/XP/Server 2003 TCP/IP stack
    - 2.3. No privileged ports
    - 2.4. Ephemeral ports allocation
    - 2.5. Identifying opened ports
        - 2.5.1. netstat command
        - 2.5.2. Identifying processes behind sockets
    - 2.6. Sockets binding and hijacking
        - 2.6.1. SO_EXCLUSIVEADDRUSE socket option
        - 2.6.2. Example of multiple bindings: NetBT driver in Windows NT 4.0 SP6a
        - 2.6.3. Multiple sockets bindings
        - 2.6.4. What happens when SO_EXCLUSIVEADDRUSE is not used?
        - 2.6.5. Windows services and drivers protected against socket hijacking
        - 2.6.6. Global protection against socket hijacking
        - 2.6.7. Diagnosing socket binding problems
    - 2.7. The missing network loopback interface
    - 2.8. Windows Vista TCP/IP stack
3. SMB/CIFS and SMB 2.0
    - 3.1. SMB/CIFS and SMB 2.0 protocols
    - 3.2. NetBIOS over TCP/IP
    - 3.3. SMB transports
    - 3.4. Vulnerabilities in Microsoft SMB/CIFS implementation
4. MSRPC, a.k.a. Microsoft implementation of DCE RPC
    - 4.1. Introduction to MSRPC
    - 4.2. DCE RPC Interface
    - 4.3. MSRPC transports
    - 4.4. MSRPC security model
    - 4.5. RPC services registration
    - 4.6. MSRPC over SMB
        - 4.6.1. Named pipes
        - 4.6.2. Named pipes used as MSRPC endpoints
        - 4.6.3. Well-known MSRPC named pipes
    - 4.7. NULL sessions
        - 4.7.1. Introduction

4.7.2. Enabling NULL sessions restrictions
4.7.3. The ANONYMOUS LOGON network logon session
4.7.4. Restrictions at the share level
4.7.5. Restrictions on named pipes (IPC$ share)
4.7.6. Hardcoded named pipes
4.7.7. Named pipes permissions
4.7.8. Named pipes firewall in Windows XP SP2, Windows Server 2003 SP1 and later versions
4.7.9. NULL sessions restrictions settings in Windows 2000
4.7.10. NULL sessions restrictions settings in Windows XP and Windows Server 2003
4.7.11. NULL session restrictions for the samr interface in Windows XP and Windows Server 2003
4.7.12. NULL session restrictions for the lsarpc interface in Windows XP and Windows Server 2003
4.7.13. NULL sessions restrictions for the samr interface on Active Directory domain contollers
4.7.14. NULL sessions restrictions for the lsarpc interface on Active Directory domain contollers
4.7.15. NULL sessions restrictions of server and workstation RPC operations
4.8. MSRPC over TCP/IP
4.8.1. Portmapper RPC service
4.8.2. RPC interfaces supported by the rpcss service
4.8.3. DCOM-related RPC interfaces running in the rpcss service
4.8.4. ORPC interfaces running in the rpcss service
4.9. Windows core MSRPC interfaces
4.9.1. lsarpc interface
4.9.2. samr interface
4.9.3. netlogon interface
4.9.4. drsuapi interface
4.9.5. dssetup interface
4.9.6. eventlog interface
4.9.7. pnp interface
4.9.8. srvsvc interface
4.9.9. svcctl interface
4.9.10. winreg interface
4.9.11. wkssvc interface
4.10. Windows services MSRPC interfaces
4.10.1. Active Directory domain controllers RPC services

4.10.2. Computer Browser service
4.10.3. DCOM Server Process Launcher
4.10.4. Distributed File System service
4.10.5. DNS server
4.10.6. Exchange RPC services
4.10.7. Exchange RPC services in Active Directory domains
4.10.8. File Replication service
4.10.9. IIS services
4.10.10. Inter-site Messaging service
4.10.11. Message Queuing and Distributed Transaction Coordinator services
4.10.12. Messenger service
4.10.13. NetDDE service
4.10.14. RPC locator service
4.10.15. Scheduler service
4.10.16. Spooler service
4.10.17. WINS service
4.11. Other MSRPC interfaces
4.11.1. Application Management service
4.11.2. Certificate services
4.11.3. Client Service for NetWare
4.11.4. Cryptographic Services service
4.11.5. DHCP Client service
4.11.6. DHCP Server service
4.11.7. Distributed Link Tracking Client service
4.11.8. Distributed Link Tracking Server service
4.11.9. DNS Client service - Windows 2000
4.11.10. DNS Client service - Windows XP and later versions
4.11.11. EFS
4.11.12. Fax server
4.11.13. File Server for Macintosh
4.11.14. IPsec Policy Agent service - Windows 2000
4.11.15. IPsec Services service - Windows XP and later versions
4.11.16. License Logging service
4.11.17. Microsoft SQL Server
4.11.18. Protected storage service
4.11.19. Routing and Remote Access service
4.11.20. Secondary Logon service
4.11.21. Security Configuration Editor Engine

4.11.22. SSDP Discovery Service service
4.11.23. System Event Notification service
4.11.24. Telephony service
4.11.25. Terminal Server service
4.11.26. WebClient service
4.11.27. Windows Audio service
4.11.28. Windows File Protection
4.11.29. Windows Security Center
4.11.30. Windows Time service
4.11.31. Winlogon process - Windows 2000
4.11.32. Winlogon process - Windows Server 2003
4.11.33. Wireless Configuration service
4.12. MSRPC interfaces introduced in Windows Vista
4.12.1. Group Policy Client Service
4.12.2. Network Location Awareness
4.12.3. Network Store Interface
4.12.4. Parental controls
4.12.5. Peer Networking Identity Manager
4.12.6. Remote Registry Service
4.12.7. Windows event collector service
4.12.8. Windows event logging service
4.12.9. Windows Firewall
4.12.10. Windows Wireless LAN 802.11 Auto Configuration Service
4.12.11. Wired Autoconfiguration Service
4.13. Implication of multiple RPC services in one process
4.13.1. Win32 services hosting
4.13.2. Example of multiple RPC services in one process
4.13.3. Implications of running multiple RPC services in one process
4.14. RPC services protection
4.15. RPC interfaces restriction in Windows XP SP2, Windows Server 2003 SP1 and later versions
4.16. MSRPC vulnerabilities
4.17. MSRPC network traffic
4.17.1. MSRPC network traffic analysis with Ethereal
4.17.2. MSRPC network traffic analysis in Network Intrusion Prevention Systems
4.17.3. MSRPC network traffic analysis in Firewalls
4.18. DCOM
4.18.1. COM interfaces

4.18.2. DCOM network traffic
5. Conclusion
Bibliography

List of Tables

4.1. MSRPC security providers
4.2. Named pipes used by MSRPC servers
4.3. epmp operations
4.4. localepmp operations
4.5. DbgIdl operations
4.6. FwIdl operations
4.7. IRemoteActivation (IActivation) operations
4.8. IOXIDResolver operations
4.9. ILocalObjectExporter operations
4.10. ISCM operations
4.11. IROT operations
4.12. IMachineActivatorControl operations
4.13. ISCMActivator operations
4.14. ISystemActivator (IRemoteSCMActivator) operations
4.15. lsarpc operations
4.16. samr operations
4.17. netlogon operations
4.18. drsuapi operations
4.19. dssetup operations
4.20. eventlog operations
4.21. pnp operations
4.22. nt4_pnp operations
4.23. srvsvc operations
4.24. svcctl operations
4.25. winreg operations
4.26. wkssvc operations
4.27. JetBack operations
4.28. JetRest operations
4.29. dsrole operations
4.30. dsaop operations
4.31. browser operations
4.32. IActivationKernel operations
4.33. netdfs operations

4.34. DnsServer operations
4.35. exchange_mapi operations
4.36. exchange_rfr operations
4.37. rxds operations
4.38. nspi operations
4.39. FrsRpc operations
4.40. NtFrsApi operations
4.41. PerfFrs operations
4.42. inetinfo operations
4.43. iis_smtp operations
4.44. iis_nntp operations
4.45. iis_imap operations
4.46. iis_pop operations
4.47. ismapi operations
4.48. ismserv_ip operations
4.49. qmcomm operations
4.50. qmcomm2 operations
4.51. qm2qm operations
4.52. qmrepl operations
4.53. qmmgmt operations
4.54. IXnRemote operations
4.55. msgsvc operations
4.56. msgsvcsend operation
4.57. nddeapi operations
4.58. NsiS operations
4.59. NsiC operations
4.60. NsiM operations
4.61. atsvc operations
4.62. sasec operations
4.63. idletask operations
4.64. ITaskSchedulerService operations
4.65. winspool operations
4.66. winsif operations
4.67. winsi2 operations
4.68. appmgmt operations
4.69. ICertPassage operations
4.70. nwwks operations
4.71. IKeySvc operations

4.72. IKeySvc2 operations
4.73. ICertProtect operations
4.74. ICatDBSvc operations
4.75. RpcSrvDHCPC operations
4.76. dhcpcsvc6 operations
4.77. dhcpsrv operations
4.78. dhcpsrv2 operations
4.79. trkwks operations
4.80. trksvr operations
4.81. dnsrslvr operations
4.82. DnsResolver operations
4.83. efsrpc operations
4.84. fax_Server operations
4.85. sfmsvc operations
4.86. PolicyAgent operations
4.87. winipsec operations
4.88. lls_license operations
4.89. llsrpc operations
4.90. RPCnetlib operations
4.91. IPStoreProv operations
4.92. ICryptProtect operations
4.93. PasswordRecovery operations
4.94. BackupKey operations
4.95. rras operations
4.96. ISeclogon operations
4.97. SceSvc operations
4.98. ssdpsrv operations
4.99. SensApi operations
4.100. SENSNotify operations
4.101. tapsrv operations
4.102. lcrpc operations
4.103. winstation_rpc operations
4.104. davclntrpc operations
4.105. AudioSrv operations
4.106. AudioSrv operations
4.107. AudioRpc operations
4.108. AudioSrv operations
4.109. sfcapi operations

4.110. SecurityCenter operations
4.111. w32time operations
4.112. InitShutdown operations
4.113. pmapapi operations
4.114. GetUserToken operations
4.115. IUserProfile operations
4.116. IProfileDialog operations
4.117. IRPCSCLogon operations
4.118. winwzc operations
4.119. IGroupPolicyUtilities operations
4.120. nlaapi operations
4.121. nlaplg operations
4.122. WinNsi operations
4.123. WPCSvc operations
4.124. IP2pIMSvc operations
4.125. IPeerGroupSvc operations
4.126. pnrpsvc operations
4.127. perflibv2 operations
4.128. ICollectorService operations
4.129. IEventService operations
4.130. FwRpc operations
4.131. Fw_Resource_Indication operations
4.132. winwlan operations
4.133. winwdiag operations
4.134. winlan operations
4.135. Vulnerabilities in MSRPC interfaces
4.136. IRemUnknown methods
4.137. IRemUnknown2 methods
4.138. IOrCallback operations

Chapter 1. Introduction

The aim of this paper is to document some not well-known characteristics of Windows systems (based on the NT kernel, i.e Windows NT, Windows 2000, Windows XP and Windows Server 2003) TCP/IP stack and network services.

The first section of the paper focuses on Windows systems TCP/IP stack, highlighting some specificities that are not well known.

The second section briefly mentions the SMB/CIFS protocol, which is probably the most important network protocol on Windows systems (not to be confused with NetBIOS over TCP/IP, as frequently seen, which is just a transport protocol for SMB/CIFS). The reference documentation for SMB/CIFS is Christopher Hertel's book, Implementing CIFS [1]

The third section deals with MSRPC, a core Windows subsystem that implements a remote procedure call method, used for local processes communication as well as remote procedures calls.

In addition to the present paper, other presentations and documents related to Windows network services are also available:

- Windows network services internals (November 2003)
- MSRPC null sessions: exploitation and protection (June 2005)
- Active Directory network protocols and traffic (May 2005)
- Active Directory network protocols and traffic (September 2004)
- Minimizing Windows Server 2003 network services (March 2005)
- Minimization of network services on Windows 2000 and XP systems (September 2002)

Chapter 2. TCP/IP stacks

Table of Contents

2.1. Introduction to Windows TCP/IP stacks
2.2. Windows 2000/XP/Server 2003 TCP/IP stack
2.3. No privileged ports
2.4. Ephemeral ports allocation
2.5. Identifying opened ports
    2.5.1. netstat command
    2.5.2. Identifying processes behind sockets
2.6. Sockets binding and hijacking
    2.6.1. SO_EXCLUSIVEADDRUSE socket option
    2.6.2. Example of multiple bindings: NetBT driver in Windows NT 4.0 SP6a
    2.6.3. Multiple sockets bindings
    2.6.4. What happens when SO_EXCLUSIVEADDRUSE is not used?
    2.6.5. Windows services and drivers protected against socket hijacking
    2.6.6. Global protection against socket hijacking
    2.6.7. Diagnosing socket binding problems
2.7. The missing network loopback interface
2.8. Windows Vista TCP/IP stack

2.1. Introduction to Windows TCP/IP stacks

The software that implements the various network protocols needed in TCP/IP environments is usually referred as a TCP/IP stack. On most systems including all versions of Windows NT, the TCP/IP stack is implemented in kernel mode. In the case of Windows NT, the **tcpip.sys** driver implements the TCP/IP stack.

All Windows NT versions up to Windows XP and Windows Server 2003 shipped with the first generation of Windows TCP/IP stack. Windows Vista and Windows Server "Longhorn" include the second generation of Windows TCP/IP stack.

2.2. Windows 2000/XP/Server 2003 TCP/IP stack

Two white papers published by Microsoft document in details the TCP/IP stack (first generation) implemented in Windows 2000, Windows XP and Windows Server 2003:

- Microsoft Windows 2000 TCP/IP Implementation Details
- Microsoft Windows Server 2003 TCP/IP Implementation Details

In addition, several articles published in the Cable Guy TechNet column [2] document with details features of the TCP/IP stack:

- December 2001: IP Routing
- January 2002: How the Windows XP Network Bridge Works
- February 2002: IP Multicast Overview
- October 2002: Understanding the IPv6 Routing Table
- June 2003: The Reliable Multicast Protocol Component of Windows Server 2003
- September 2003: Default Gateway Behavior for Windows TCP/IP
- January 2004: New Networking Features in Microsoft Windows XP Service Pack 2
- July 2004: Path Maximum Transmission Unit (PMTU) Black Hole Routers
- December 2004: New Networking Features in Microsoft Windows Server 2003 Service Pack 1
- October 2005: TCP/IP Packet Processing Paths
- December 2005: Windows TCP/IP Ephemeral, Reserved, and Blocked Port Behavior

2.3. No privileged ports

Unix systems implement privileged ports: ports lower than 1024 can only be used by the system administrator (root user). Considering that typical internet servers run on a low TCP port (for example, 25/tcp for an SMTP server or 80/tcp for an HTTP server), this limitation ensures that only the system administrator can run such servers.

Windows systems do not implement privileged ports. As a consequence, anybody can bind a TCP or UDP server on a low port. As explained later, this has some serious security implications.

2.4. Ephemeral ports allocation

This section was written prior to the publication of the Windows TCP/IP Ephemeral, Reserved, and Blocked Port Behavior Microsoft article, which contains a detailed explanation of how TCP and UDP ports are used in Windows XP and Windows Server 2003.

In the TCP/IP model, dynamic ports are typically used as source port by a TCP or UDP client, to communicate with a remote TCP or UDP server, using a well-known port as destination port. In Windows systems, dynamic ports are also used by RPC services (in that case, a portmapper service is needed to find the appropriate RPC service).

When an application or driver requests a dynamic TCP or UDP port from the TCP/IP driver, the allocated port belongs by default to the 1025-5000 range (port 1024 is apparently never used on Windows systems).

The upper limit of this range can be changed, modifying the following registry value:

```
Key: HKLM\SYSTEM\CCS\Services\TcpIp\Parameters\
Value: MaxUserPort (REG_DWORD)
Default value: 5000 (decimal)
```

This range is shared for TCP and UDP ports. Moreover, dynamic ports are allocated incrementally. For example, if an application requests a TCP port and obtains TCP port 1025, the next application requesting a UDP port will obtain port 1026.

Exclusion from the dynamic port range can be configured with the ReservedPorts registry value:

```
Key: HKLM\SYSTEM\CCS\Services\TcpIp\Parameters\
Value: ReservedPorts (REG_MULTI_SZ)
```

Configuring this value can be necessary when some services need a fixed port in the lower part of the dynamic range, like 1080/tcp for a SOCKS proxy or 1433/tcp and 1434/udp for MS SQL Server. Otherwise, such ports may be dynamically allocated before services startup, which would cause the service start failure.

However, it seems that the ReservedPorts registry value is also used by the Windows 2000 IPv4 NAT

driver [4], to determine which range can be used for source ports of NATed connections.

2.5. Identifying opened ports

2.5.1. netstat command
2.5.2. Identifying processes behind sockets

2.5.1. netstat command

Systems implementing the TCP/IP protocol typically include the netstat utility, which can be used, among other things, to list opened sockets.

The netstat command of Windows systems is known to be buggy:

- Before Windows NT 4.0 SP3, netstat does not display listening TCP ports [5]
- On Windows NT 4.0, netstat displays TCP ports as listening, when sockets are only bound to UDP ports [6]

The second bug can lead to suprising netstat outputs on Windows NT 4.0 systems. One particularly odd result is that TCP port 135 (used by the rpcss service, as explained later) is displayed twice in netstat outputs:

```
C:\WINNT>netstat -anp tcp | find ":135 "
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  UDP    0.0.0.0:135            *:*
```

This is because the rpcss service opens both ports 135/tcp and 135/udp. But, with the bug aforementionned, 135/tcp is displayed a second time. This explains why 135/tcp appears twice.

Another serious bug exists in all versions of Windows NT systems before Windows Server 2003 and Windows XP SP2: for each outgoing TCP connection established from a Windows system, the local source port is displayed as LISTENING [7].

In the following example, a TCP connection was established to port 22 of a remote server. The TCP/IP driver allocated port 1367 as source port for the connection. In the netstat output, the port appears in the LISTENING state:

```
C:\WINDOWS>netstat -anp tcp | find ":1367"
  TCP    0.0.0.0:1367           0.0.0.0:0              LISTENING
  TCP    192.70.106.142:1367    192.70.106.76:22       ESTABLISHED
```

However, this port is not really in the LISTENING state, i.e, it is not possible to establish a new TCP connection on port 1367. Using hping [8] to send a TCP segment with the SYN flag set, a TCP segment with the RST-ACK flags set is returned:

```
jbm@garbarek ~> sudo hping -S -c 1 192.70.106.142 -p 1367
HPING 192.70.106.142 (ep1 192.70.106.142): S set, 40 headers + 0 data bytes
len=46 ip=192.70.106.142 flags=RA seq=0 ttl=127 id=47511 win=0 rtt=3.7 ms

--- 192.70.106.142 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 3.7/3.7/3.7 ms
```

It turns out that this bug comes from an incorrect mapping between TDI objects and TCP sockets.

The Winsock API (implementation of BSD sockets API on Windows systems) is implemented on TCP/IP using the Afd driver, which uses the TDI (Transport Driver Interface) API to communicate with the TCP/IP driver.

To implement an outgoing TCP connection, the Afd driver creates two TDI objets:

- a TDI address object
- a TDI connection object

Using a simple TCP client (nc.exe, [9]) to establish a TCP connection to port 22 of a remote server:

```
C:\WINNT>nc -z 192.168.1.254 22
```

the implementation at the TDI level can be monitored, using the TDIMon tool [10]:

```
1  8246D3F0 IRP_MJ_CREATE            TCP:0.0.0.0:0      SUCCESS Address Open
2  8246D3F0 TDI_SET_EVENT_HANDLER    TCP:0.0.0.0:1038   SUCCESS Error Event
3  8246D3F0 TDI_SET_EVENT_HANDLER    TCP:0.0.0.0:1038   SUCCESS Disconnect
Event
4  8246D3F0 TDI_SET_EVENT_HANDLER    TCP:0.0.0.0:1038   SUCCESS Receive Event
5  8246D3F0 TDI_SET_EVENT_HANDLER    TCP:0.0.0.0:1038   SUCCESS Expedited Receive
Event
6  8246D3F0 TDI_SET_EVENT_HANDLER    TCP:0.0.0.0:1038   SUCCESS Chained Receive
Event
7  8246D3F0 TDI_QUERY_INFORMATION    TCP:0.0.0.0:1038   SUCCESS Query Address
8  824C1AE0 IRP_MJ_CREATE            TCP:Connection obj SUCCESS
Context:0x822CF9B8
9  824C1AE0 TDI_ASSOCIATE_ADDRES     TCP:Connection obj SUCCESS
TCP:0.0.0.0:1038
10 824C1AE0 TDI_CONNECT              TCP:0.0.0.0:1038   192.168.1.254:22
SUCCESS
```

The output can be interpreted as follow:

- line 1, a create request (**IRP_MJ_CREATE**) for a TDI address object is sent to the TCP/IP driver. The drivers returns an object with 0x8246D3F0 address.
- from line 2 to line 6, handlers are associated with the object, for the different events that can occur. In particular, line 4 associates a handler to receive notifications when data arrive on port 1038.
- line 8, 9 and 10 show the creation of the TDI object used to represent the outgoing TCP connection. On line 10, the **TDI_CONNECT** command establishes the TCP connection to port 22 of the machine with 192.168.1.254 as IP address.

Thus, it appears that at the TDI level, a TCP connection is implemented using two TDI objects:

- one object representing the TCP connection itself
- one object used to receive data sent to the local port

The problem is that the **GetTcpTable()** API retrieving the content of the current TCP connections table incorrectly

translates the second TDI object as a TCP listening socket. As a consequence, the port is displayed as LISTENING by the netstat command. Note that any tools using this API will report incorrect results. Thus, results of such tools must be analyzed carefully, to filter ports reported as LISTENING. This bug has been fixed in Windows Server 2003 and in Windows XP SP2.

2.5.2. Identifying processes behind sockets

Starting with Windows XP, the netstat command can be used with the **-o** option to identify which process opened a given socket [12]. Starting with Windows XP SP2 and Windows Server 2003 SP1, the **-b** option can be used instead of the **-o** option.

In October 2005, Microsoft documented the availability of a Windows 2000 update [13], adding support for the **-o** netstat option.

On systems where the **-o** netstat option is not available, the following tools can be used:

- Sysinternal's TCPView (GUI) and tcpvcon (command-line interface) [14]
- Fport [16]

These tools will give the PID (Process Identifier) of processes using sockets.

However, knowing the PID is not always enough to identify precisely which system component opened a given socket, particularly in the following cases:

- Standard Windows services run in a few shared processes (**services.exe**, **svchost.exe**). The aforementionned tools return the PID of the process but can not identify which service in a shared process opened a given socket. It is then necessary to stop services inside the shared process, to determine which service owns a given socket.
- Some sockets are reported as owned by the System process.

On a default Windows system, some sockets will be reported as owned by the System process (pid 8 on Windows 2000, pid 4 on Windows XP and Windows Server 2003): these sockets are opened by drivers communicating directly with the TCP/IP driver in kernel-mode.

It is not possible to statically identify which driver opened a given port. Thus, it is sometimes hard to figure out why a port is opened when it has been opened by a driver. For example, on some Windows systems, port 1025 (the first dynamic port) seems to be opened by an unknown driver at system startup.

The following well-known ports are opened by the following drivers:

- 137/udp, 138/udp, 139/tcp, 445/tcp, 445/udp: **netbt.sys**

- source ports used for outgoing SMB sessions (with a TCP destination port equal to 139/tcp or 445/tcp): **netbts.sys**
- 1701/udp, 1723/tcp: **raspptp.sys**

For more information, a list of TCP and UDP ports used by Microsoft Server Products is available [15].

2.6. Sockets binding and hijacking

2.6.1. SO_EXCLUSIVEADDRUSE socket option
2.6.2. Example of multiple bindings: NetBT driver in Windows NT 4.0 SP6a
2.6.3. Multiple sockets bindings
2.6.4. What happens when SO_EXCLUSIVEADDRUSE is not used?
2.6.5. Windows services and drivers protected against socket hijacking
2.6.6. Global protection against socket hijacking
2.6.7. Diagnosing socket binding problems

As explained earlier, Windows TCP/IP stack does not implement privileged ports. More precisely, any process can bind a socket to any port, even when a socket is already bound to a port. Thus, it becomes possible to hijack a TCP server.

This kind of vulnerability was published for the first time in february 1998, in the security advisory NT port binding security [17].

This advisory showed how, for example, any user could hijack the Windows NT 4 SMB server, binding a TCP server on port TCP 139 using a specific IP address in the **bind()** call.

Microsoft released knowledge base article 194431 [21], mentionning the problem and stating that it was fixed in Windows NT 4.0 Service Pack 4.

Actually, Microsoft introduced in NT 4.0 Service Pack 4 a new socket option, **SO_EXCLUSIVEADDRUSE**, that can be used by an application to protect itself from this vulnerability. However:

- it seems that Microsoft itself did not use this socket option in its servers (particularly, IIS 4 and IIS 5)
- this socket option can not be used by drivers, which directly communicate with the TCP/IP driver, without using the Winsock API.

2.6.1. SO_EXCLUSIVEADDRUSE socket option

The **SO_EXCLUSIVEADDRUSE** socket option is documented as follow in MSDN [22]:

> The SO_EXCLUSIVEADDRUSE option prevents other sockets from being forcibly bound to the same address and port, a practice enabled by the SO_REUSEADDR option; such reuse can be executed by malicious applications to disrupt the application.

Thus, when this socket option is used by an application before using the **bind()** function, no other application will be able to bind to the same local address, even when the **SO_REUSEADDR** is used, as does nc.exe.

As said earlier, the Winsock API is implemented by the Afd driver, which interacts with the TCP/IP driver using the TDI interface. At the TDI level, TCP and UDP ports are represented by file objects.

The implementation of the **SO_EXCLUSIVEADDRUSE** socket option opens file objects in exclusive mode, setting the **ShareAccess** parameter of the **ZwCreateFile()** function to 0. Thus, file objects representing TCP and UDP ports can only be opened in exclusive mode, which correspond to exclusive binding at the Winsock level.

Warning: before Windows 2000 SP4, Windows XP SP2 or Windows Server 2003, this socket option can only be used by processes running with administrator credentials. This bug is documented in the #870562 Microsoft knowledge base article [23].

2.6.2. Example of multiple bindings: NetBT driver in Windows NT 4.0 SP6a

Follows a demonstration of multiple bindings on a Windows NT 4.0 SP6a system. As NetBIOS over TCP/IP is active on the system, TCP Port 139 is opened by the NetBT driver and bound to IP address 192.70.106.143:

```
C:\>netstat -an | find "139"
  TCP    192.70.106.143:139     0.0.0.0:0              LISTENING
```

Then, a **nc.exe** process is bound to the same port and same IP address:

```
C:\>nc -l -p 139 -s 192.70.106.143

C:\>netstat -an | find "139"
  TCP    192.70.106.143:139     0.0.0.0:0              LISTENING
  TCP    192.70.106.143:139     0.0.0.0:0              LISTENING
```

The next TCP connection will be routed to the **nc.exe** process, hijacking the SMB server.

Using socat [18] to establish a TCP connection to port 139 of IP address 192.70.106.143, the **blah** string is sent:

```
jbm@garbarek ~> socat - tcp4:192.70.106.143:139
blah
```

The **blah** string is received by the **nc.exe** process.

```
C:\>nc -l -p 139 -s 192.70.106.143
blah

C:\>
```

An interesting way to exploit this vulnerability would be to setup an SMB redirector, that would redirect all SMB trafic to another machine [19].

When Microsoft introduced the **SO_EXCLUSIVEADDRUSE** socket option in Windows NT 4.0

Service Pack 4, it did not fixed that problem because the NetBT driver was not modified to set the **ShareAccess** parameter of **ZwCreateFile()** functions calls to 0.

A fix for the NetBT driver was finally introduced in the C2 Update Post-SP6a hotfix, because one TCSEC C2 requirement mandates that an unprivileged user-mode program should not be able to listen to ports used by Windows NT services [20].

This fix is also available in the Windows NT 4.0 Security Rollup Package. To enable it, the following registry value must be configured:

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\NetBT\Parameters
Value: EnablePortLocking (REG_DWORD)
Content: 0 to disable protection (default), 1 to enable protection
```

Prev | Up | Next

2.6.1. SO_EXCLUSIVEADDRUSE socket option | Home | 2.6.3. Multiple sockets bindings

2.6.3. Multiple sockets bindings

Considering TCP servers, there are different case of multiple sockets bindings, that can occur when the first server did not specify **SO_EXCLUSIVEADDRUSE** and when the second server specifies **SO_REUSEADDR** is used by the second server

- One TCP server bound to all interfaces (**INADDR_ANY** or **0.0.0.0**) and then, a second TCP server bound to a specific interface
- One TCP server bound to a specific interface and then, a second TCP server bound to all interfaces
- One TCP server bound to a specific interface and then, a second TCP server to another specific interface
- One TCP server bound to a specific interface and then, a second TCP server bound to the same specific interface

The first case is a serious security problem. This means that if a TCP server is bound to all interfaces, it is later possible to start a TCP server bound to the same port but on a specific interface. The second TCP server will receive all TCP connection segments sent to the IP adress of the specific interface.

As the TCP/IP stack does not implement privileged ports, it is possible to disrupt any TCP servers using this technique.

The second case is not a security problem. The second server will receive TCP connection segments sent to any IP address different from the IP address of the specific interface.

The third case is not a security problem, as the two servers are listening on different specific interfaces.

The fourth case is problematic because two TCP servers are bound to exactly the same local address (same port and same IP address). The MSDN documentation [22] explains that in that case, the behavior is undefined as to which sockets will receive incoming connection requests.

However, it seems that on Windows NT 4.0, the second server will receive packets, which is the worst case because this means that the first server is hijacked. This is what happens with the NeBT driver on Windows NT 4.0 SP6a, as seen earlier.

As a conclusion, it seems important to use the **SO_EXCLUSIVEADDRUSE** socket option to prevent sockets hijacking.

2.6.4. What happens when SO_EXCLUSIVEADDRUSE is not used?

Even if Microsoft introduced the **SO_EXCLUSIVEADDRUSE** socket option in Windows NT 4.0 Service Pack 4, it seems that it was not used in some Microsoft application servers.

For instance, the HTTP server part of Internet Information Services (IIS) 5, shipped with Windows 2000, listens by default on all network interfaces on ports 80 and 443. It is possible to hijack the HTTP server of IIS 5 with a TCP server bound to the IP address of a specific interface.

Even more interesting, when a TCP server listens on all interfaces, it is possible to silently intercept TCP traffic, binding a second TCP server to intercept trafic and redirecting to the loopback address, to finally deliver date to the hijacked server (thanks to Franck Davy for suggesting this).

On a Windows 2000 server with IIS 5, the HTTP service listens on all interfaces:

```
C:\WINNT\>netstat -an | find "80"
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
```

Using fpipe, a second TCP server is bound to IPv4 address 192.70.106.142 and configured to redirect traffic to the loopback address (127.0.0.1), with TCP port 80 as destination:

```
C:\WINNT\>fpipe -l 80 -r 80 -i 192.70.106.142 127.0.0.1 -v
FPipe v2.1 - TCP/UDP port redirector.
Copyright 2000 (c) by Foundstone, Inc.
http://www.foundstone.com

Listening for TCP connections on 192.70.106.142 port 80
```

Using socat to send an HTTP query:

```
jbm@garbarek ~> socat - tcp4:192.70.106.142:80
GET / HTTP/1.1
Host: localhost
```

The second server receives the connection on port 80 and redirect data to the IIS 5 server, using 127.0.0.1 as destination address:

```
Connection accepted from 192.70.106.76 port 1077
Attempting to connect to 192.70.106.76 port 1077
Pipe connected:
   In:    192.70.106.76:1077  --> 192.70.106.142:80
  Out:   192.70.106.142:33014 --> 127.0.0.1:80
15 bytes received from inbound connection
16 bytes received from inbound connection
1 bytes received from inbound connection
273 bytes received from outbound connection
Outbound connection lost
Closing outbound connection
Closing inbound connection

Quit signal detected. Shutting down...
```

The TCP client finally receives data sent by the IIS5 server:

```
HTTP/1.1 404 Object Not Found
Server: Microsoft-IIS/5.0
Date: Thu, 22 May 2003 16:25:32 GMT
Connection: close
Content-Type: text/html
Content-Length: 111

[...]
```

2.6.5. Windows services and drivers protected against socket hijacking

Most Windows 2000 (and later Windows NT versions) network services are protected with the **SO_EXCLUSIVEADDRUSE** socket option:

- All RPC services that use TCP sockets (135/tcp, dynamic TCP ports in range 1025-5000) apparently use the socket option
- MS SQL Server 2000 (1433/tcp)

All ports opened by Windows 2000 drivers (and later Windows NT version) correctly set the **ShareAccess** parameter to 0 when calling **ZwCreateFile()**;

- NetBT driver: 137/udp, 138/udp, 139/tcp, 445/tcp
- PPTP driver: 1723/tcp

2.6.6. Global protection against socket hijacking

As explained earlier, to be protected against socket hijacking, applications must explicitly set the **SO_EXCLUSIVEADDRUSE** socket option before calling **bind()**.

Thus, applications must be modified to be protected, which is typically not possible.

A registry value exists under the **Parameters** key of the Afd driver, to globally enable the protection, as if all applications specified the socket option:

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\Afd\Parameters\
Value: DisableAddressSharing
Content: 1 (to enable protection)
```

2.6.7. Diagnosing socket binding problems

Diagnosing socket binding problems is easy when the return code of the bind() call is available. The typical return codes are:

- WSAEADDRINUSE (10048): this typically means that the **SO_REUSEADDR** socket option was not used.
- WSAEACCES (10013): this means that socket hijacking protection was enabled (either with **SO_EXCLUSIVEADDRUSE** or with **DisableAddressSharing** set to 1.

Most of the times, the bind() return code is not available. In that case, the TDIMon [10] tool can be used:

```
1       0.00000000      my_netcat.exe:65        819D38B8        IRP_MJ_CREATE
TCP:0.0.0.0:445         SHARING_VIOLATION       Address Open
```

- When the WSAEADDRINUSE error is returned, TDIMon does not display anything, because this error code is returned directly by the Afd driver, without ever communicating with the TCP/IP driver.
- When the WSAEACCES error is returned, TDIMon displays the **SHARING_VIOLATION** error, returned at the TDI level

2.7. The missing network loopback interface

Windows TCP/IP stack does not implement a network loopback interface, as found in other TCP/IP stack like **lo*** interfaces in BSD systems.

Thus, it is not possible to sniff network traffic using the typical Windows packet capture driver, WinPcap [24].

However, TDIMon [10] can be used to see loopback traffic (but does not support network traffic logging):

- Sending process
- Source and destination ports (source and destination IP address are typically 127.0.0.1)
- Length of sent data

NT Kernel Resources's Local Host Monitor [11] is similar to Sysinternals's TDIMon but also supports data sent to the TCP/IP driver using the TDI interface, including loopback network data.

Another method to capture loopback network data is to use a Winsock LSP (Layered Service Provider). The Ports Traffic Analyzer [25] is an example of such sniffer.

The Microsoft Loopback Adapter can be installed on Windows systems, to run network applications when no physical adapter is present or active on the system [26]. This adapter is not the equivalent of a network loopback interface and IPv4 address 127.0.0.1 can not be assigned to it. Also, it is not possible to sniff network traffic on it, at least with WinPcap.

2.8. Windows Vista TCP/IP stack

Changes to networking features in Windows Vista and Windows Server "Longhorn" are described in the New Networking Features in Windows Server "Longhorn" and Windows Vista Technet article.

Several issues of the Cable Guy Technet column document with more details features of the new stack:

- September 2005: Next Generation TCP/IP Stack in Windows Vista and Windows Server "Longhorn"
- October 2005: Changes to IPv6 in Windows Vista and Windows Server "Longhorn"
- November 2005: Performance Enhancements in the Next Generation TCP/IP Stack
- January 2006: The New Windows Firewall in Windows Vista and Windows Server "Longhorn"
- March 2006: Policy-based QoS Architecture in Windows Server "Longhorn" and Windows Vista
- April 2006: Connecting to Wireless Networks with Windows Vista
- May 2006: Configuring IPv6 with Windows Vista

Chapter 3. SMB/CIFS and SMB 2.0

Table of Contents

3.1. SMB/CIFS and SMB 2.0 protocols
3.2. NetBIOS over TCP/IP
3.3. SMB transports
3.4. Vulnerabilities in Microsoft SMB/CIFS implementation

3.1. SMB/CIFS and SMB 2.0 protocols

The SMB (Server Message Block) protocol, renamed at some point CIFS (Common Internet File System), is the protocol behind resource sharing and remote administration functionnalities in Windows systems. It is implemented in all Windows NT systems.

For a thorough explanation of the SMB/CIFS protocol, see the SMB chapter [27] of Christopher Hertel's book, Implementing CIFS.

SMB 2.0 is the new version of SMB implemented in Windows Vista. For more information about SMB 2.0, see the SMB 2.0 pages on the Ethereal Wiki.

3.2. NetBIOS over TCP/IP

NetBIOS over TCP/IP uses 3 ports:

- 137/UDP, for NetBIOS name resolution, using broadcast or a WINS server
- 138/UDP, for session-less NetBIOS
- 139/TCP, for session-oriented NetBIOS

The NetBIOS API uses names to identify network ressources. The **nbtstat** command can be used to examine and configure NetBIOS names on a Windows system.

NetBIOS over TCP/IP in itself is just a transport protocol. However, as it is the typical transport protocol for the SMB/CIFS protocol in Windows systems, it is often confused with the SMB/CIFS protocol, which does the real work on Windows systems.

For technical details about NetBIOS over TCP/IP, the reference documentation is the NBT chapter [28] of Christopher Hertel's book, Implementing CIFS.

3.3. SMB transports

Before Windows 2000, the typical transport protocol of SMB/CIFS was NetBIOS over TCP/IP. Starting with Windows 2000, SMB/CIFS can be carried directly into TCP (445/tcp), without an intermediary NetBT layer.

To identify which SMB transports are active on a Windows system, the **net config rdr** and **net config srv** commands can be used. These commands use the **NetWkstaTransportEnum()** and **NetServerTransportEnum()** Win32 API:

```
C:\WINNT>net config rdr

[...]

Workstation active on
        NetbiosSmb (000000000000)
        NetBT_Tcpip_{33227EBB-55A3-49EA-823D-51836B978EFD} (000102A495B2)

[...]

C:\WINNT>net config srv

[...]

Server is active on
        NetBT_Tcpip_{33227EBB-55A3-49EA-823D-51836B978EFD} (000102a495b2)
        NetBT_Tcpip_{33227EBB-55A3-49EA-823D-51836B978EFD} (000102a495b2)
        NetbiosSmb (000000000000)
        NetbiosSmb (000000000000)

[...]
```

The **NetWkstaTransportEnum()** and **NetServerTransportEnum()** Win32 API are implemented by two RPC calls, **NetrWkstaTransportEnum()** and **NetrServerTransportEnum()**. Samba-TNG [29] rpcclient utility supports the **srvtransports** command, that can be used to retrieve server-side transports.

Note: Windows NT 4.0 and Windows 2000 systems apparently have a bug in the **NetServerTransportEnum()** API, which retrieves server-side transports: each transport appears twice.

In Windows Vista, the output of the **net config srv** is as follows:

```
C:\WINDOWS>net config srv

[...]

Software version                        Windows (TM) Code Name "Longhor
Server is active on
        NetbiosSmb (WINVISTA)
        NetBT_TCPIP_{34559422-6B8D-4328-BAA1-25A6A331C6A8} (WINVISTA)

[...]
```

Active transports are:

- **NetbiosSmb** is the raw SMB transport (445/tcp) [30].
- **NetBT_Tcpip_{...}** is the NetBT SMB transport, bound on a per-adapter basis

The raw SMB transport can not be disabled on a per-adapter basis. To completely disable it, the NetBT driver must be stopped.

A Windows system with both SMB transports active tries to connect to 445/tcp and 139/tcp at the same time. If the connection to 445/tcp is accepted, the connection to port 139 is closed (sending a TCP segment with the RST flag set), i.e., raw SMB transport is preferred over NetBT transport [31].

3.4. Vulnerabilities in Microsoft SMB/CIFS implementation

- MS02-045: Unchecked Buffer in Network Share Provider Can Lead to Denial of Service (CVE-2002-0724)
- MS02-070: Flaw in SMB Signing Could Enable Group Policy to be Modified (CVE-2002-1256)
- MS03-005: Unchecked buffer in Windows redirector may permit privilege elevation (CVE-2003-0004)
- MS05-027: Vulnerability in Server Message Block Could Allow Remote Code Execution (CVE-2005-1206)

Chapter 4. MSRPC, a.k.a. Microsoft implementation of DCE RPC

Table of Contents

4.1. Introduction to MSRPC
4.2. DCE RPC Interface
4.3. MSRPC transports
4.4. MSRPC security model
4.5. RPC services registration
4.6. MSRPC over SMB
    4.6.1. Named pipes
    4.6.2. Named pipes used as MSRPC endpoints
    4.6.3. Well-known MSRPC named pipes
4.7. NULL sessions
    4.7.1. Introduction
    4.7.2. Enabling NULL sessions restrictions
    4.7.3. The ANONYMOUS LOGON network logon session
    4.7.4. Restrictions at the share level
    4.7.5. Restrictions on named pipes (IPC$ share)
    4.7.6. Hardcoded named pipes
    4.7.7. Named pipes permissions
    4.7.8. Named pipes firewall in Windows XP SP2, Windows Server 2003 SP1 and later versions
    4.7.9. NULL sessions restrictions settings in Windows 2000
    4.7.10. NULL sessions restrictions settings in Windows XP and Windows Server 2003
    4.7.11. NULL session restrictions for the samr interface in Windows XP and Windows Server 2003
    4.7.12. NULL session restrictions for the lsarpc interface in Windows XP and Windows Server 2003
    4.7.13. NULL sessions restrictions for the samr interface on Active Directory domain contollers
    4.7.14. NULL sessions restrictions for the lsarpc interface on Active Directory domain contollers
    4.7.15. NULL sessions restrictions of server and workstation RPC operations
4.8. MSRPC over TCP/IP
    4.8.1. Portmapper RPC service
    4.8.2. RPC interfaces supported by the rpcss service

4.8.3. DCOM-related RPC interfaces running in the rpcss service
4.8.4. ORPC interfaces running in the rpcss service
4.9. Windows core MSRPC interfaces
4.9.1. lsarpc interface
4.9.2. samr interface
4.9.3. netlogon interface
4.9.4. drsuapi interface
4.9.5. dssetup interface
4.9.6. eventlog interface
4.9.7. pnp interface
4.9.8. srvsvc interface
4.9.9. svcctl interface
4.9.10. winreg interface
4.9.11. wkssvc interface
4.10. Windows services MSRPC interfaces
4.10.1. Active Directory domain controllers RPC services
4.10.2. Computer Browser service
4.10.3. DCOM Server Process Launcher
4.10.4. Distributed File System service
4.10.5. DNS server
4.10.6. Exchange RPC services
4.10.7. Exchange RPC services in Active Directory domains
4.10.8. File Replication service
4.10.9. IIS services
4.10.10. Inter-site Messaging service
4.10.11. Message Queuing and Distributed Transaction Coordinator services
4.10.12. Messenger service
4.10.13. NetDDE service
4.10.14. RPC locator service
4.10.15. Scheduler service
4.10.16. Spooler service
4.10.17. WINS service
4.11. Other MSRPC interfaces
4.11.1. Application Management service
4.11.2. Certificate services
4.11.3. Client Service for NetWare
4.11.4. Cryptographic Services service
4.11.5. DHCP Client service

4.11.6. DHCP Server service
4.11.7. Distributed Link Tracking Client service
4.11.8. Distributed Link Tracking Server service
4.11.9. DNS Client service - Windows 2000
4.11.10. DNS Client service - Windows XP and later versions
4.11.11. EFS
4.11.12. Fax server
4.11.13. File Server for Macintosh
4.11.14. IPsec Policy Agent service - Windows 2000
4.11.15. IPsec Services service - Windows XP and later versions
4.11.16. License Logging service
4.11.17. Microsoft SQL Server
4.11.18. Protected storage service
4.11.19. Routing and Remote Access service
4.11.20. Secondary Logon service
4.11.21. Security Configuration Editor Engine
4.11.22. SSDP Discovery Service service
4.11.23. System Event Notification service
4.11.24. Telephony service
4.11.25. Terminal Server service
4.11.26. WebClient service
4.11.27. Windows Audio service
4.11.28. Windows File Protection
4.11.29. Windows Security Center
4.11.30. Windows Time service
4.11.31. Winlogon process - Windows 2000
4.11.32. Winlogon process - Windows Server 2003
4.11.33. Wireless Configuration service
4.12. MSRPC interfaces introduced in Windows Vista
4.12.1. Group Policy Client Service
4.12.2. Network Location Awareness
4.12.3. Network Store Interface
4.12.4. Parental controls
4.12.5. Peer Networking Identity Manager
4.12.6. Remote Registry Service
4.12.7. Windows event collector service
4.12.8. Windows event logging service
4.12.9. Windows Firewall

4.12.10. Windows Wireless LAN 802.11 Auto Configuration Service
4.12.11. Wired Autoconfiguration Service
4.13. Implication of multiple RPC services in one process
4.13.1. Win32 services hosting
4.13.2. Example of multiple RPC services in one process
4.13.3. Implications of running multiple RPC services in one process
4.14. RPC services protection
4.15. RPC interfaces restriction in Windows XP SP2, Windows Server 2003 SP1 and later versions
4.16. MSRPC vulnerabilities
4.17. MSRPC network traffic
4.17.1. MSRPC network traffic analysis with Ethereal
4.17.2. MSRPC network traffic analysis in Network Intrusion Prevention Systems
4.17.3. MSRPC network traffic analysis in Firewalls
4.18. DCOM
4.18.1. COM interfaces
4.18.2. DCOM network traffic

Prev

Next

3.4. Vulnerabilities in Microsoft SMB/ CIFS implementation

Home

4.1. Introduction to MSRPC

4.1. Introduction to MSRPC

The RPC (Remote Procedure Call) mechanism allows an application to seamlessly invoke remote procedures, as if these procedures were executed locally.

There are two main implementations of the RPC mechanism:

- ONC RPC [32]
- DCE RPC [33]

MSRPC is the Microsoft implementation of the DCE RPC mechanism. In particular, Microsoft added new transport protocols for DCE RPC, in particular the **ncacn_np** transport, which use named pipes carried into the SMB protocol.

For an interesting story of Windows and how Microsoft chose to implement DCE RPC, see A brief history of Windows [34].

For a general overview of the MSRPC architecture, see the RPC Technical Reference, What Is RPC? and How RPC Works sections in the Windows Server 2003 Technical Library.

4.2. DCE RPC Interface

An interface is a set of related operations (procedures) that can be invoked remotely. Each interface is distinguished by an interface identifier (ifid) and an interface version number.

For a detailed explanation of DCE RPC interface, see figure 2.2 [35] in the DCE RPC 1.1 documentation.

4.3. MSRPC transports

The RPC mechanism was designed to be transport-independant: different protocols can be used to transport remote procedure parameters and execution results.

In DCE-RPC (and thus MSRPC), transport protocols are identified with protocol sequences identifiers. Windows systems typically use the following protocol sequences:

- **ncacn_ip_tcp**: TCP/IP transport
- **ncadg_ip_udp**: UDP/IP transport
- **ncacn_np**: named pipes transport, using SMB
- **ncalrpc**: local RPC
- **ncacn_http**: HTTP transport, using IIS

An endpoint is the entity used at the transport level to invoke remotely a RPC service. Endpoint nature is specific to each protocol sequences:

- **ncacn_ip_tcp**: TCP port
- **ncadg_ip_udp**: UDP port
- **ncacn_np**: named pipe
- **ncalrpc**: LPC port
- **ncacn_http**: 593/tcp

Most LPC ports are MSRPC endpoints. Using the Winobj tool [36], you can see a list of LPC ports used as MSRPC endpoints on a running system, under the **RPC Control** subdirectory of the NT kernel Object Manager namespace.

However, not all TCP or UDP ports are MSRPC endpoints, as well as not all named pipes.

One method to identify if a TCP port, UDP port or named pipe is a MSRPC endpoint is to try to bind to the RPC service supposedly listening on the supposed endpoint. If the bind operation fails or blocks, then the tested endpoint is probably not a MSRPC endpoint.

The ifids tool, part of Todd Sabin's RPC Tools [37] can be used to identify RPC services endpoints. A demonstration of this tool is given in [41].

4.4. MSRPC security model

MSRPC uses the Windows SSPI (Security Support Provider Interface) to use security services such as authentication and confidentiality.

The list of MSRPC security providers is stored under the following registry key:

```
Key: HKLM\SOFTWAWRE\Microsoft\Rpc\SecurityService\
```

The following MSRPC security providers are defined:

Table 4.1. MSRPC security providers

Security Provider	Integer	DLL
DCE private key authentication	1	secur32.dll
SPNEGO	9	secur32.dll
NTLM	10	secur32.dll
Schannel (SSL, PCT, TLS)	14	schannel.dll
MS Kerberos	16	secur32.dll
MSN SSP	17	
Distributed Password Authentication	18	secur32.dll
Netlogon secure channel	68	netlogon.dll
Microsoft Message Queue (MSMQ)	100	

When the SMB transport (ncacn_np) is used, there is no additional authentication at the MSRPC level. Instead, the security context of the MSRPC session is derived from the authenticated SMB session established previously.

4.5. RPC services registration

When a RPC service starts, it can register its endpoints along with the interface identifier and version of the service. A special RPC service, the portmapper service, maintains a database, the endpoint map, that can be queried to find out endpoints that can be used to invoke a given RPC service.

When a RPC service listens on a TCP or UDP endpoint, it must register itself in the endpoint map because TCP and UDP ports are dynamically allocated to RPC services.

To query the portmapper RPC service, the rpcdump tool [37] can be used. In the output of that command, **ncacn_ip_tcp** and **ncadg_ip_udp** correspond to dynamically allocated ports.

4.6. MSRPC over SMB

4.6.1. Named pipes
4.6.2. Named pipes used as MSRPC endpoints
4.6.3. Well-known MSRPC named pipes

4.6.1. Named pipes

In Windows systems, named pipes in one of the available IPC (Inter-Process Communication) mechanism. It can be used either locally or remotely.

Accesses to remote named pipes, contained in the IPC$ share, are carried into the SMB protocol.

Named pipes are implemented by a file system driver, **npfs.sys**. The PipeList [38] tool can be used to enumerate the npfs namespace, to show which named pipes are opened on a local system. The FileMon tool [39] is also able to monitor the named pipe filesystem activity, by selecting Named Pipes in the Drives menu.

Some named pipes are implemented as aliases [40], i.e, they don't really exist in the npfs namespace. Aliases names are stored in the registry:

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\Npfs\Aliases\
Values: lsass, ntsvcs
```

Named pipes are protected by security descriptors, just like any Windows NT objects. The pipeacl tool ([42],[43]) be used to examine the content of security descriptors protecting named pipes.

4.6.2. Named pipes used as MSRPC endpoints

Some named pipes are used as MSRPC endpoints, i.e, they are used to carry DCE RPC PDU (Protocol Data Units). Compared to other RPC transports, the **ncacn_np** transport is different because it usually does not involve authentication at the DCE RPC level. Instead, authentication information is obtained from the SMB session established priorly.

However, it is possible to have unauthenticated DCE RPC sessions using SMB NULL sessions. In that case, there is no authentication, neither at the SMB level nor at the DCE RPC level.

Also, contrary to RPC services listening on **ncacn_ip_tcp** or **ncadg_ip_udp**, the portmapper service is generally not necessary for RPC services using named pipes, as the named pipe name usually identifies the RPC interface.

4.6.3. Well-known MSRPC named pipes

The following table gives a list of named pipes that are used as endpoints by Windows RPC services. The interface identifier associated to each named pipe represents the service typically accessed when a given named pipe is used.

However, due to the fact that any endpoint in a given process can be used to reach any RPC service, it is possible to use multiple RPC services using a single named pipe endpoint.

Table 4.2. Named pipes used by MSRPC servers

Named pipe	Description	Service or process	Interface identifier
atsvc	atsvc interface (Scheduler service)	mstask.exe	1ff70682-0a51-30e8-076d-740be8cee98b v1.0
AudioSrv	AudioSrv interface (Windows Audio service)	AudioSrv	3faf4738-3a21-4307-b46c-fdda9bb8c0d5 v1.0
browser (ntsvcs alias)	browser interface (Computer Browser service)	Browser	6bffd098-a112-3610-9833-012892020162 v0.0
cert	ICertPassage interface (Certificate services)	certsrv.exe	91ae6020-9e3c-11cf-8d7c-00aa00c091be v0.0
Ctx_Winstation_API_Service	winstation_rpc interface	termsrv.exe	5ca4a760-ebb1-11cf-8611-00a0245420ed v1.0
DAV RPC SERVICE	davclntrpc interface (WebDAV client service)	WebClient	c8cb7687-e6d3-11d2-a958-00c04f682e16 v1.0
dnsserver	DnsServer interface (DNS Server service)	dns.exe	50abc2a4-574d-40b3-9d66-ee4fd5fba076 v5.0

epmapper	epmp interface (RPC endpoint mapper)	RpcSs	e1af8308-5d1f-11c9-91a4-08002b14a0fa v3.0
eventlog (ntsvcs alias)	eventlog interface (Eventlog service)	Eventlog	82273fdc-e32a-18c3-3f78-827929dc23ea v0.0
HydraLsPipe	Terminal Server Licensing	lserver.exe	3d267954-eeb7-11d1-b94e-00c04fa3080d v1.0
InitShutdown	InitShutdown interface	winlogon.exe	894de0c0-0d55-11d3-a322-00c04fa321a1 v1.0
keysvc	IKeySvc interface (Cryptographic services)	CryptSvc	8d0ffe72-d252-11d0-bf8f-00c04fd9126b v1.0
keysvc	ICertProtect interface (Cryptographic services)	CryptSvc	0d72a7d4-6148-11d1-b4aa-00c04fb66ea0 v1.0
locator	NsiS interface (RPC Locator service)	locator.exe	d6d70ef0-0e3b-11cb-acc3-08002b1d29c4 v1.0
llsrpc	llsrpc interface (Licensing Logging service)	llssrv.exe	342cfd40-3c6c-11ce-a893-08002b2e9c6d v0.0
lsarpc (lsass alias)	lsarpc interface	lsass.exe	12345778-1234-abcd-ef00-0123456789ab v0.0
lsarpc (lsass alias)	dssetup interface	lsass.exe	3919286a-b10c-11d0-9ba8-00c04fd92ef5 v0.0
msgsvc (ntsvcs alias)	msgsvcsend interface (Messenger service)	messenger	5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc v1.0
nddeapi	nddeapi interface (NetDDE service)	netdde.exe	2f5f3220-c126-1076-b549-074d078619da v1.2

netdfs	netdfs interface (Distributed File System service)	Dfssvc	4fc742e0-4a10-11cf-8273-00aa004ae673 v3.0
netlogon (lsass alias)	netlogon interface (Net Logon service)	Netlogon	12345678-1234-abcd-ef00-01234567cffb v1.0
ntsvcs	pnp interface (Plug and Play service)	PlugPlay	8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0
plugplay	pnp interface (Plug and Play Windows Vista service)	PlugPlay	8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0
policyagent	PolicyAgent interface (IPSEC Policy Agent (Windows 2000))	PolicyAgent	d335b8f6-cb31-11d0-b0f9-006097ba4e54 v1.5
ipsec	winipsec interface (IPsec Services)	PolicyAgent	12345678-1234-abcd-ef00-0123456789ab v1.0
ProfMapApi	pmapapi interface	winlogon.exe	369ce4f0-0fdc-11d3-bde8-00c04f8eee78 v1.0
protected_storage	IPStoreProv interface (Protected Storage)	lsass.exe	c9378ff1-16f7-11d0-a0b2-00aa0061426a v1.0
ROUTER	Remote Access	mprdim.dll	8f09f000-b7ed-11ce-bbd2-00001a181cad v0.0
samr (lsass alias)	samr interface	lsass.exe	12345778-1234-abcd-ef00-0123456789ac v1.0
scerpc	SceSvc	services.exe	93149ca2-973b-11d1-8c39-00c04fb984f9 v0.0
SECLOGON	ISeclogon interface (Secondary logon service)	seclogon	12b81e99-f207-4a4c-85d3-77b42f76fd14 v1.0

SfcApi	sfcapi interface (Windows File Protection)	winlogon.exe	83da7c00-e84f-11d2-9807-00c04f8ec850 v2.0
spoolss	spoolss interface (Spooler service)	spoolsv.exe	12345678-1234-abcd-ef00-0123456789ab v1.0
srvsvc (ntsvcs alias)	srvsvc interface (Server service)	services.exe (w2k) or svchost.exe (wxp and w2k3)	4b324fc8-1670-01d3-1278-5a47bf6ee188 v3.0
ssdpsrv	ssdpsrv interface (SSDP service)	ssdpsrv	4b112204-0e19-11d3-b42b-0000f81feb9f v1.0
svcctl (ntsvcs alias)	svcctl interface (Services control manager)	services.exe	367aeb81-9844-35f1-ad32-98f038001003 v2.0
tapsrv	tapsrv interface (Telephony service)	Tapisrv	2f5f6520-ca46-1067-b319-00dd010662da v1.0
trkwks	trkwks interface (Distributed Link Tracking Client)	Trkwks	300f3532-38cc-11d0-a3f0-0020af6b0add v1.2
W32TIME (ntsvcs alias)	w32time interface (Windows Time (Windows 2000 and XP))	w32time	8fb6d884-2388-11d0-8c35-00c04fda2795 v4.1
W32TIME_ALT	w32time interface (Windows Time (Windows Server 2003, Windows Vista))	w32time	8fb6d884-2388-11d0-8c35-00c04fda2795 v4.1
winlogonrpc	GetUserToken interface	winlogon.exe	a002b3a0-c9b7-11d1-ae88-0080c75e4ec1 v1.0
winreg	winreg interface (Remote registry service)	RemoteRegistry	338cd001-2244-31f1-aaaa-900038001003 v1.0

winspipe	winsif interface (WINS service)	wins.exe	45f52c28-7f9f-101a-b52b-08002b2efabe v1.0
wkssvc (ntsvcs alias)	wkssvc interface (Workstation service)	services.exe (w2k) or svchost.exe (wxp and w2k3)	6bffd098-a112-3610-9833-46c3f87e345a v1.0

4.7. NULL sessions

4.7.1. Introduction
4.7.2. Enabling NULL sessions restrictions
4.7.3. The ANONYMOUS LOGON network logon session
4.7.4. Restrictions at the share level
4.7.5. Restrictions on named pipes (IPC$ share)
4.7.6. Hardcoded named pipes
4.7.7. Named pipes permissions
4.7.8. Named pipes firewall in Windows XP SP2, Windows Server 2003 SP1 and later versions
4.7.9. NULL sessions restrictions settings in Windows 2000
4.7.10. NULL sessions restrictions settings in Windows XP and Windows Server 2003
4.7.11. NULL session restrictions for the samr interface in Windows XP and Windows Server 2003
4.7.12. NULL session restrictions for the lsarpc interface in Windows XP and Windows Server 2003
4.7.13. NULL sessions restrictions for the samr interface on Active Directory domain contollers
4.7.14. NULL sessions restrictions for the lsarpc interface on Active Directory domain contollers
4.7.15. NULL sessions restrictions of server and workstation RPC operations

4.7.1. Introduction

NULL sessions refer to the possibility to use unauthenticated SMB sessions to the IPC$ share to gather information anonymously, using RPC function calls carried into SMB.

SMB sessions are typically authenticated. However, it is possible to use an empty username and password, which results in a NULL session, i.e an anonymous SMB session.

The MSRPC NULL sessions: exploitation and protection presentation is available to complement the information found in this section.

4.7.2. Enabling NULL sessions restrictions

In Windows NT, NULL sessions were used by processes running under the LOCALSYSTEM logon session when they needed to establish a network logon session on a remote server.

Because processes running as LOCALSYSTEM did not have network credentials in Windows NT, the only way to establish a network logon session on a remote server was to use an empty login and password during SMB authentication.

It turned out that NULL sessions could be used to gather some sensible information anonymously.

Microsoft added NULL sessions restrictions in Windows NT 3.5. These restrictions are enabled by the following registry value, which is enabled by default starting with NT 3.5:

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\
Value: RestrictNullSessAccess (REG_DWORD)
Content: 1 to enable NULL sessions restrictions (default value)
```

In recent Windows systems, this registry value (enabled by default) is also a security option:

> Network access: Restrict anonymous access to Named Pipes and Shares

The different NULL sessions restrictions are detailed in the next sections.

4.7.3. The ANONYMOUS LOGON network logon session

When a Windows systems starts, it creates several logon sessions (depending on Windows versions), including a network logon session to represent anonymous access. This network logon session is known as the **ANONYMOUS LOGON** network logon session.

Logon sessions created at startup by a Windows Server 2003 can be enumerated using the logonsessions tool [44].

```
Z:\>logonsessions

Logonsesions v1.1
Copyright (C) 2004 Bryce Cogswell and Mark Russinovich
Sysinternals - wwww.sysinternals.com


[0] Logon session 00000000:000003e7:
    User name:    WORKGROUP\W2K3DFLT$
    Auth package: NTLM
    Logon type:   (none)
    Session:      0
    Sid:          S-1-5-18
    Logon time:   21/01/2005 03:49:29
    Logon server:
    DNS Domain:
    UPN:

[1] Logon session 00000000:00006e13:
    User name:
    Auth package: NTLM
    Logon type:   (none)
    Session:      0
    Sid:          (none)
    Logon time:   21/01/2005 03:49:29
    Logon server:
    DNS Domain:
    UPN:
```

```
[2] Logon session 00000000:000003e5:
    User name:    NT AUTHORITY\LOCAL SERVICE
    Auth package: Negotiate
    Logon type:   Service
    Session:      0
    Sid:          S-1-5-19
    Logon time:   21/01/2005 03:49:30
    Logon server:
    DNS Domain:
    UPN:

[3] Logon session 00000000:000003e4:
    User name:    NT AUTHORITY\NETWORK SERVICE
    Auth package: Negotiate
    Logon type:   Service
    Session:      0
    Sid:          S-1-5-20
    Logon time:   21/01/2005 03:49:30
    Logon server:
    DNS Domain:
    UPN:

[4] Logon session 00000000:0000e0cc:
    User name:    NT AUTHORITY\ANONYMOUS LOGON
    Auth package: NTLM
    Logon type:   Network
    Session:      0
    Sid:          S-1-5-7
    Logon time:   21/01/2005 03:49:39
    Logon server:
    DNS Domain:
    UPN:

[5] Logon session 00000000:00010a42:
    User name:    W2K3DFLT\Administrator
    Auth package: NTLM
    Logon type:   Interactive
    Session:      0
    Sid:          S-1-5-21-2330557087-2467616270-843640848-500
    Logon time:   21/01/2005 03:49:48
    Logon server: W2K3DFLT
    DNS Domain:
    UPN:
```

A Windows Server 2003 system creates 5 logon sessions at system startup:

- **LOCALSYSTEM** logon session (LUID 0x3e7)
- **LOCAL SERVICE** service logon session (LUID 0x3e5) and **NETWORK SERVICE** service logon session (LUID 0x3e4) (only in Windows XP and Windows Server 2003)
- **ANONYMOUS LOGON** network logon session
- one unknown and apparently unused logon session ([1] in the output example)

Because the **ANONYMOUS LOGON** network logon session is created at startup, when a NULL session is established, Windows does not need to create another logon session.

Hence, logon rights are not verified and it is not possible to prevent NULL sessions by removing the network logon right for **ANONYMOUS LOGON**, as one might expect.

Prev | Up | Next

4.7.2. Enabling NULL sessions restrictions | Home | 4.7.4. Restrictions at the share level

4.7.4. Restrictions at the share level

The first category of restrictions allows the administrator to configure which shares can be used anonymously:

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\
Value: NullSessionShares (REG_SZ)
```

This registry value is also set by a security option, starting with Windows XP:

> Network access: Shares that can be accessed anonymously

On default Windows systems, this value contains the **COMCFG** and **DFS$** shares.

The **IPC$** share does not appear in this registry value. However, it is always possible to connect anonymously to it. Restrictions for the **IPC$** share are implemented at the named pipes level, as described in the next section.

4.7.5. Restrictions on named pipes (IPC$ share)

The **NullSessionPipes** registry value is supposed to configure the list of named pipes that can be opened anonymously:

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\
Value: NullSessionPipes (REG_SZ)
```

On default Windows NT 4.0 systems, the list of named pipes is:

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\
Value: NullSessionPipes (REG_SZ)
Default value: COMNAP COMNODE SQL\QUERY SPOOLSS LLSRPC EPMAPPER LOCATOR WINREG
```

On default Windows 2000 systems, there are two more named pipes (**TrkWks** and **TrkSvr**, opened by the Distributed Link Tracking Client and Distributed Link Tracking Server services):

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\
Value: NullSessionPipes (REG_SZ)
Default value: COMNAP COMNODE SQL\QUERY SPOOLSS LLSRPC EPMAPPER LOCATOR TrkWks TrkSvr
```

In Windows 2000 Service Pack 4 (SP4), the **LLSRPC** named pipe was removed, as documented by Microsoft [45]. Note however that **LLSRPC** is removed only when SP4 is installed on an existing Windows 2000 installation, but NOT when SP4 is slipstreamed to create a Windows 2000 SP4 installation image.

Starting with Windows XP, this registry value can be set via a security option:

> Network access: Pipes that can be accessed anonymously

4.7.6. Hardcoded named pipes

In addition to named pipes that appear in the **NullSessionPipes** registry value, some additional named pipes are hardcoded in the SMB server driver (**srv.sys**).

The following named pipes are hardcoded:

```
\pipe\lsarpc, \pipe\samr, \pipe\netlogon (\pipe\lsass aliases)
\pipe\wkssvc, \pipe\srvsvc, \pipe\browser (\pipe\ntsvcs aliases)
```

Thus, it is possible to open the **lsarpc** named pipe in the context of a NULL session (but not the **lsass** named pipe, even if the first one is an alias of the second one, as explained earlier).

These harcoded named pipes were removed in the SMB server driver of recent Windows systems, starting with Windows XP Service Pack 2 (SP2) and Windows Server 2003 Service Pack 1 (SP1).

On these systems, the **NullSessionPipes** registry value was updated.

On Windows XP SP2, **browser** (no longer hardcoded) was explicitely added:

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\
Value: NullSessionPipes (REG_SZ)
Default value: COMNAP, COMNODE, SQL\QUERY, SPOOLSS, LLSRPC, browser
```

As a consequence, it is no longer possible to open the following named pipes in the context of a NULL session in Windows XP SP2:

- **\pipe\lsarpc**
- **\pipe\samr**
- **\pipe\netlogon**
- **\pipe\wkssvc**
- **\pipe\srvsvc**

On Windows Server 2003 SP1, **netlogon**, **lsarpc**, **samr** and **browser** have been explicitely added:

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\
Value: NullSessionPipes (REG_SZ)
Default value: COMNAP, COMNODE, SQL\QUERY, SPOOLSS, netlogon, lsarpc, samr, browser
```

As a consequence, it is no longer to open the following named pipes in the context of a NULL session in Windows Server 2003 SP1:

- **\pipe\wkssvc**
- **\pipe\srvsvc**

4.7.7. Named pipes permissions

Named pipes are implemented by a filesystem driver in Windows NT, npfs.sys, which supports security descriptors on each named pipe. These security descriptors are used to control access to named pipes. It is possible to use the pipeacl tool ([42], [43]) to examine and modify security descriptors on named pipes.

In Windows 2000, named pipes DACL (Discretionnary Access Control Lists) grant permissions to EVERYONE and ADMINISTRATORS for most named pipes used as MSRPC endpoints. Because ANONYMOUS LOGON is included in EVERYONE in Windows 2000, named pipes permissions allow anonymous accesses.

```
E:\>pipeacl \??\pipe\lsarpc
Revision: 1
Reserved: 0
Control : 8004
Owner: BUILTIN\Administrators (S-1-5-32-544)
Group: SYSTEM (S-1-5-18)
Sacl: Not present
Dacl: 2 aces
(A) (00) 0012019b :     Everyone (S-1-1-0)
(A) (00) 001f01ff :     BUILTIN\Administrators (S-1-5-32-544)
```

In Windows XP and Windows Server 2003, DACL grant permissions to EVERYONE, ANONYMOUS LOGON and ADMINISTRATORS. EVERYONE and ANONYMOUS LOGON are given exactly the same permissions: permissions are thus exactly equivalent to Windows 2000 permissions (starting with Windows XP, EVERYONE does not include ANONYMOUS LOGON so ANONYMOUS LOGON must explicitely appear in ACL).

```
C:\>pipeacl \??\pipe\lsarpc
Revision: 1
Reserved: 0
Control : 8004
Owner: BUILTIN\Administrators (S-1-5-32-544)
Group: SYSTEM (S-1-5-18)
Sacl: Not present
Dacl: 3 aces
```

```
(A) (00) 001f01ff :     BUILTIN\Administrators (S-1-5-32-544)
(A) (00) 0012019b :     Anonymous (S-1-5-7)
(A) (00) 0012019b :     Everyone (S-1-1-0)
```

It is possible to modify ACL on named pipes using pipeacl and typically either add a deny ACE for ANONYMOUS LOGON in Windows 2000 or remove the ACE for ANONYMOUS LOGON in Windows XP and Windows Server 2003.

When permissions are manually removed for ANONYMOUS LOGON for named pipes that are either hardcoded in the SMB server driver or found in the **NullSessionPipes** registry value such as **lsarpc**, it is no longer possible to open this pipe in the context of a NULL session.

To conclude, permissions on named permissions are typically not used for NULL sessions restrictions because, by default, DACL allow accesses for ANONYMOUS LOGON. It is not really practical to modify default DACL, given that modifications of DACL on named pipes are not persistent (because named pipes are created by RPC services at startup).

Prev | Up | Next

4.7.6. Hardcoded named pipes | Home | 4.7.8. Named pipes firewall in Windows XP SP2, Windows Server 2003 SP1 and later versions

4.7.8. Named pipes firewall in Windows XP SP2, Windows Server 2003 SP1 and later versions

A named pipes filtering feature was introduced in Windows XP SP2 and Windows Server 2003 SP1.

Named pipes filtering can be dynamically enabled (without requiring a reboot) by adding and setting the following registry value to 1:

```
Key: HKLM\SYSTEM\CCS\Services\lanmanserver\Parameters\
Value: PipeFirewallActive (REG_DWORD)
Content: 1 to enable named pipe filtering
```

The list of allowed named pipes can then be dynamically configured (named pipes can be added or removed without requiring a reboot) in the following registry value:

```
Key: HKLM\SYSTEM\CCS\Services\lanmanserver\Parameters\
Value: AllowedPipes (REG_MULTI_SZ)
Content: list of allowed named pipes
```

When **PipeFirewallActive** is set to 1 and **AllowedPipes** is empty, accesses to any named pipe is refused, both for unauthenticated and authenticated sessions.

4.7.9. NULL sessions restrictions settings in Windows 2000

Actually, NULL sessions have security implications because the security context of a NULL session contains the EVERYONE SID. Thus, the EVERYONE group includes anonymous users and, if a DACL allows some accesses for the EVERYONE group, such accesses can be executed in the context of a NULL session. Microsoft introduced the AUTHENTICATED USERS group in Windows NT 4.0 SP3, that contains only authenticated users. This group can be used to grant permissions instead of EVERYONE.

Also, starting with Windows NT 4.0 SP3, the LSA (Local Security Authority) can be configured to restrict the capabilities of a NULL session, with the following registry value:

```
Key: HKLM\SYSTEM\CurrentControlSet\Control\LSA\
Value: RestrictAnonymous
Content: 0 (no restriction), 1 (some restrictions), 2 (only valid in Windows 2000)
```

This registry value is also a group policy security option, starting with Windows 2000:

```
Additional restrictions for anonymous connections: None. Rely on default permissions
(RestrictAnonymous == 0)
Additional restrictions for anonymous connections: Do not allow enumeration of SAM
accounts and shares (RestrictAnonymous == 1)
Additional restrictions for anonymous connections: No access without explicit
anonymous permissions (RestrictAnonymous == 2)
```

Setting **RestrictAnonymous** to 2 completely disables NULL sessions, by removing the **EVERYONE** SID from the token of a NULL session.

When **RestrictAnonymous** is set to 1 in Windows NT or Windows 2000, it is still possible to gather some interesting information anonymously [48], using the appropriate functions calls and tools [49].

4.7.10. NULL sessions restrictions settings in Windows XP and Windows Server 2003

Starting with Windows XP, EVERYONE does not contain ANONYMOUS LOGON, because the following security option, which sets the **EveryoneIncludesAnonymous** registry value, is disabled by default:

> Network access: Let Everyone permissions apply to anonymous users (Disabled by default)

As a consequence, when permissions must allow anonymous accesses, they explicitely grant access to ANONYMOUS LOGON.

The **RestrictAnonymous** registry value still exists in Windows XP and Windows Server 2003 and corresponds to the following security option:

> Network access: Do not allow anonymous enumeration of SAM accounts and shares (Disabled by default)

Given that this security option can either be disabled or enabled, it is easy to deduce that the only valid values for the **RestrictAnonymous** registry value in Windows XP and Windows Server 2003 are 0 or 1 (2 is not supported, contrary to Windows 2000).

Because of the **EveryoneIncludesAnonymous** registry value default setting in Windows XP and Windows Server 2003, **RestrictAnonymous** is not as important as it was in Windows 2000.

4.7.11. NULL session restrictions for the samr interface in Windows XP and Windows Server 2003

Windows XP and Windows Server 2003 (systems that are not Active Directory domain controllers) have one security option that can be used to either block (by default) or allow all anonymous bind to the **samr** interface:

> Network access: Do not allow anonymous enumeration of SAM accounts (Enabled by default)

This security option sets or unsets the **RestrictAnonymousSam** registry value to 1. Because this option is set by default on Windows XP and Windows Server 2003, it is not possible to connect anonymously to the SAM server on Windows XP and Windows Server 2003 (except for Windows Server 2003 domain controllers).

In practice, all calls to **SamrConnect*** operations fail with permission denied.

4.7.12. NULL session restrictions for the lsarpc interface in Windows XP and Windows Server 2003

Windows XP and Windows Server 2003 have one security option (enabled by default) that disables or enables the anonymous translation of SID to name:

> Network access: Allow anonymous SID/Name translation (Disabled by default)

When the first security option is enabled, the DACL on the LSA policy object is modified, as shown with the lsaacl tool [50]:

- When the option is disabled, an ACE explicitly denies the Lookup Names access for the ANONYMOUS LOGON SID
- When it is set, this ACE is modified, to allow the View Local Info and Lookup Names accesses for the ANONYMOUS LOGON SID.

This change is dynamic and does not require a reboot.

In Windows XP, anonymous connections to the **lsarpc** interface are allowed by default but because the aforementionned option is disabled by default, it it not possible to translate anonymously SID to name.

On Windows Server 2003 systems that are not Active Directory domain controllers, anonymous connections to the **lsarpc** interface are forbidden by default.

In practice, all calls to LsarOpenPolicy or LsarOpenPolicy2 operations fail on non-DC Windows Server 2003 systems.

Thus, the value of the aforementionned option (disabled or enabled) typically does not matter on non-DC Windows Server 2003 systems, except if the **TurnOffAnonymousBlock** registry value was explictly added and set to 1.

4.7.13. NULL sessions restrictions for the samr interface on Active Directory domain contollers

On Active Directory domain controllers, NULL sessions restrictions for the **samr** interface are based on members of the **Pre-Windows 2000 Compatible Access** group, because this group is used in DACL of Active Directory objects.

Because of the default setting proposed by **dcpromo.exe**, this group typically contains:

- **EVERYONE** on Windows 2000 Active Directory domain controllers (**EVERYONE** includes anonymous users in Windows 2000)
- **ANONYMOUS LOGON** on Windows Server 2003 Active Directory domain controllers

As a consequence, anonymous accesses to the **samr** interface are typically possible on Active Directory domain controllers, including full enumeration of accounts stored in AD.

Modification of the **Pre-Windows 2000 Compatible Access** group requires a reboot.

4.7.14. NULL sessions restrictions for the lsarpc interface on Active Directory domain contollers

On both Windows 2000 and Windows Server 2003 Active Directory domain controllers, it is possible to connect anonymously to the **lsarpc** interface.

In Windows Server 2003, this is because the **TurnOffAnonymousBlock** registry value is added and set to 1 when a Windows Server 2003 server is promoted to an Active Directory domain controller.

Note: the modification of the **TurnOffAnonymousBlock** registry value in Windows Server 2003 does not require a reboot.

On Windows 2000 servers (including Active Directory domain controllers), anonymous translation of SID to name is allowed, even if the **RestrictAnonymous** registry value is set to 1.

On Windows Server 2003 servers (including Active Directory domain controllers), the following security option

> Network access: Allow anonymous SID/Name translation

forbids or allows anonymous translation of SID to name.

4.7.15. NULL sessions restrictions of server and workstation RPC operations

For some of the lanmanserver and lanmanworkstation RPC services operations (**srvsvc** and **wkssvc** named pipes), restrictions are hardcoded and documented in MSDN, under the Security requirements section. Sometimes, depending on the requested information level, it is necessary (or not) to be a member of the Administrators or Account Operators local group.

The following **srvsvc** operations can be used anonymously:

- NetrShareEnum (levels 1 and 2 only)
- NetrServerTransportEnum

In addition, on Windows 2000 workstation and member servers, the following **srvsvc** operations can be used anonymously if **RestrictAnonymous** is set to 0:

- NetrServerGetInfo (levels 100 and 101 only)

The following **wkssvc** operations can be used anonymously:

- NetrWkstaGetInfo (level 100 only)
- NetrWkstaTransportEnum

It is possible to modify the security requirements for some of the **srvsvc** operations, modifying some of the security descriptors found under the **DefaultSecurity** registry key, under the **lanmanserver** registry key.

On a default Windows 2000 system, the following registry values are available:

- SrvsvcConfigInfo
- SrvsvcFile
- SrvsvcServerDiskEnum
- SrvsvcSessionInfo
- SrvsvcShareAdminConnect
- SrvsvcShareAdminInfo
- SrvsvcShareConnect
- SrvsvcShareFileInfo
- SrvsvcSharePrintInfo

- SrvsvcStatisticsInfo

On Windows XP and Windows Server 2003, additional security descriptors exist:

- SrvsvcConnection
- SrvsvcTransportEnum

The Tweak UI tool (part of Microsoft PowerToys for Windows XP) has an Access Control feature that allows the configuration of these security descriptors for Windows XP and Windows Server 2003:

- Manage file and printer sharing (SrvsvcConfigInfo)
- Manage file/print server connections (SrvsvcConnection)
- Manage file server open files (SrvsvcFile)
- Enumerate file servers disks (SrvsvcServerDiskEnum)
- Manage file/print server sessions (SrvsvcSessionInfo)
- Connect to administrative shares (SrvsvcShareAdminConnect)
- Manage administrative shares (SrvsvcShareAdminInfo)
- Connect to file and printer shares (SrvsvcShareConnect)
- Manage file shares (SrvsvcShareFileInfo)
- Manage printer shares (SrvsvcSharePrintInfo)
- Read file/print server statistics (SrvsvcStatisticsInfo)
- Enumerate server transport protocols (SrvsvcTransportEnum)

Using Tweak UI, it is possible to harden Windows XP and Windows Server 2003 against NULL sessions to the **srvsvc** interface, removing ACE that contain ANONYMOUS LOGON.

The security descriptors are only read when the lanmanserver service starts. Thus, any modification requires a restart of the service.

4.8. MSRPC over TCP/IP

4.8.1. Portmapper RPC service
4.8.2. RPC interfaces supported by the rpcss service
4.8.3. DCOM-related RPC interfaces running in the rpcss service
4.8.4. ORPC interfaces running in the rpcss service

4.8.1. Portmapper RPC service

TCP/IP RPC services listen on dynamic TCP or UDP ports. Thus, to reach a given RPC service, identified by its interface identifier (UUID), a port mapping service is necessary.

The portmapper service is an RPC service listening on different endpoints:

- **ncalrpc**: epmapper LPC port
- **ncacn_np**: epmapper named pipe
- **ncacn_ip_tcp**: 135/tcp
- **ncadg_ip_udp**: 135/udp
- **ncacn_http**: 593/tcp

Typically, to discover the port on which a given RPC service can be reached, a client will establish a TCP connection to port 135, asking for the port allocated to a given RPC service. Then, the client closes the connection to port 135 and opens a new connection to the port returned by the portmapper service.

To register itself in the endpoint database maintained by the portmapper service, a service calls the **RpcEpRegister()** function.

By default, TCP/IP ports for RPC services are allocated in the range of dynamic ports, which starts at 1025. This explains why on most Windows systems, ports immediately higher than 1024 are used by RPC services. It is possible to configure a specific ports range for RPC services, using the rpccfg tool, as described in another document [68].

To query the portmapper service, it is possible to use a tool typically named rpcdump. Microsoft resource kit contains a Windows version of rpcdump. There is also a Windows version in Todd Sabin's RPC Tools [37], whereas Dave Aitel's SPIKE toolkit contains dcedump [69], a version running on Unix.

Using ifids on one of the portmapper RPC service endpoints, it appears that different RPC interfaces are supported on a Windows 2000 machine:

```
C:\> ifids -p ncacn_np -e \pipe\epmapper \\.
Interfaces: 11
  e1af8308-5d1f-11c9-91a4-08002b14a0fa v3.0
  0b0a6584-9e0f-11cf-a3cf-00805f68cb1b v1.1
  975201b0-59ca-11d0-a8d5-00a0c90d8051 v1.0
```

```
  e60c73e6-88f9-11cf-9af1-0020af6e72f4 v2.0
  99fcfec4-5260-101b-bbcb-00aa0021347a v0.0
  b9e79e60-3d52-11ce-aaa1-00006901293f v0.2
  412f241e-c12a-11ce-abff-0020af6e7a17 v0.2
  00000136-0000-0000-c000-000000000046 v0.0
  c6f3ee72-ce7e-11d1-b71e-00c04fc3111a v1.0
  4d9f4ab8-7d1c-11cf-861e-0020af6e7c57 v0.0
  000001a0-0000-0000-c000-000000000046 v0.0
```

On a Windows XP or Windows Server 2003 system, the result is:

```
C:\WINDOWS> ifids -p ncacn_ip_tcp -e 135 127.0.0.1
Interfaces: 11
  e1af8308-5d1f-11c9-91a4-08002b14a0fa v3.0
  0b0a6584-9e0f-11cf-a3cf-00805f68cb1b v1.1
  1d55b526-c137-46c5-ab79-638f2a68e869 v1.0
  e60c73e6-88f9-11cf-9af1-0020af6e72f4 v2.0
  99fcfec4-5260-101b-bbcb-00aa0021347a v0.0
  b9e79e60-3d52-11ce-aaa1-00006901293f v0.2
  412f241e-c12a-11ce-abff-0020af6e7a17 v0.2
  00000136-0000-0000-c000-000000000046 v0.0
  c6f3ee72-ce7e-11d1-b71e-00c04fc3111a v1.0
  4d9f4ab8-7d1c-11cf-861e-0020af6e7c57 v0.0
  000001a0-0000-0000-c000-000000000046 v0.0
```

On a Windows Vista system, there are 12 interfaces:

```
C:\WINDOWS> ifids -p ncacn_np -e epmapper \\.
Interfaces: 12
  e1af8308-5d1f-11c9-91a4-08002b14a0fa v3.0
  0b0a6584-9e0f-11cf-a3cf-00805f68cb1b v1.1
  1d55b526-c137-46c5-ab79-638f2a68e869 v1.0
  e60c73e6-88f9-11cf-9af1-0020af6e72f4 v2.0
  99fcfec4-5260-101b-bbcb-00aa0021347a v0.0
  b9e79e60-3d52-11ce-aaa1-00006901293f v0.2
  412f241e-c12a-11ce-abff-0020af6e7a17 v0.2
  00000136-0000-0000-c000-000000000046 v0.0
  c6f3ee72-ce7e-11d1-b71e-00c04fc3111a v1.0
  4d9f4ab8-7d1c-11cf-861e-0020af6e7c57 v0.0
  000001a0-0000-0000-c000-000000000046 v0.0
  64fe0b7f-9ef5-4553-a7db-9a1975777554 v1.0
```

As explained later, some of these interfaces are supposed to be only used locally whereas some are designed to be used remotely. However, because all these RPC services run in the same process, they appear when querying one endpoint of the rpcss service such as TCP port 135 or **epmapper** named pipe.

These RPC interface identifiers are classified and explained in the next sections.

4.8.2. RPC interfaces supported by the rpcss service

Note: names and purposes of some of the interfaces described in the three following sections have been documented by Microsoft France technical departments.

The first RPC interface is the DCE RPC endpoint portmapper interface [70]:

`e1af8308-5d1f-11c9-91a4-08002b14a0fa v3.0: epmp`

Table 4.3. epmp operations

Interface	Operation number	Operation name
e1af8308-5d1f-11c9-91a4-08002b14a0fa v3.0: epmp		
	0x00	ept_insert
	0x01	ept_delete
	0x02	ept_lookup
	0x03	ept_map
	0x04	ept_lookup_handle_free
	0x05	ept_inq_object
	0x06	ept_mgmt_delete
	0x07	ept_map_auth
	0x08	ept_map_auth_async

The **ept_map_auth** and **ept_map_auth_async** operations are apparently specific to Microsoft implementation of the epmapper interface.

The second RPC interface is used by local processes to reach the local endpoint mapper:

`0b0a6584-9e0f-11cf-a3cf-00805f68cb1b v1.1: localepmp`

Table 4.4. localepmp operations

Interface	Operation number	Operation name
0b0a6584-9e0f-11cf-a3cf-00805f68cb1b v1.1: localepmp		
	0x00	OpenEndpointMapper
	0x01	AllocateReservedIPPort
	0x02	ept_insert_ex
	0x03	ept_delete_ex
> Windows XP SP2	0x04	SetRestrictRemoteClients
	0x05	ResetWithNoAuthException

Starting with Windows XP, a new RPC interface is available, DbgIdl, to help debugging of RPC services:

```
1d55b526-c137-46c5-ab79-638f2a68e869 v1.0: DbgIdl
```

Table 4.5. DbgIdl operations

Interface	Operation number	Operation name
1d55b526-c137-46c5-ab79-638f2a68e869 v1.0: DbgIdl		
	0x00	GetCellByDebugCellID
	0x01	OpenRPCDebugCallInfoEnumeration
	0x02	GetNextRPCDebugCallInfo
	0x03	FinishRPCDebugCallInfoEnumeration
	0x04	OpenRPCDebugEndpointInfoEnumeration
	0x05	GetNextRPCDebugEndpointInfo
	0x06	FinishRPCDebugEndpointInfoEnumeration
	0x07	OpenRPCDebugThreadInfoEnumeration
	0x08	GetNextRPCDebugThreadInfo
	0x09	FinishRPCDebugThreadInfoEnumeration
	0x0a	OpenRPCDebugClientCallInfoEnumeration
	0x0b	GetNextRPCDebugClientCallInfo
	0x0c	FinishRPCDebugClientCallInfoEnumeration

To use the DbgIdl interface, the sytem hosting RPC services must be configured to maintain RPC troubleshooting state information. The **RPC Troubleshooting State Information** GPO must be enabled, typically in the LGPO using **gpedit.msc**.

The **dbgrpc.exe** program, available in the Microsoft Debugging tools package, can then be used, either locally or remotely to debug RPC services. More information about the usage of **dbgrpc.exe** is available in the RPC Debugging section of the Microsoft Debugging Tools package documentation [72].

The **FwIdl** interface was introduced in Windows Vista:

```
Z:\WINDOWS> ifids -p ncalrpc -e epmapper vista
Interfaces: 12

[...]

  64fe0b7f-9ef5-4553-a7db-9a1975777554 v1.0
```

Table 4.6. FwIdl operations

Interface	Operation number	Operation name
64fe0b7f-9ef5-4553-a7db-9a1975777554 v1.0: FwIdl		
	0x00	FwConnectToManager
	0x01	FwSubscribeForNewRulesNotification
	0x02	FwInterfaceRegistered
	0x03	FwPortRegistered
	0x04	FwPortUnregistered

4.8.3. DCOM-related RPC interfaces running in the rpcss service

The rpcss service not only runs the RPC subsystem but also the COM Service Control Manager (SCM), which is at the core of the COM/DCOM infrastructure. As a result, some RPC services are available in the rpcss service, as well as some ORPC services, as explained in the next section.

The IRemoteActivation (IActivation) interface is an RPC interface implemented by the COM SCM (Services Control Manager) to handle COM objects activation requests :

```
4d9f4ab8-7d1c-11cf-861e-0020af6e7c57 v0.0: IRemoteActivation (IActivation)
```

The IRemoteActivation RPC interface has exactly one operation, **RemoteActivation()**, as described in section 6.2 of the DCOM specification Section 4.18, “DCOM”.

Table 4.7. IRemoteActivation (IActivation) operations

Interface	Operation number	Operation name
4d9f4ab8-7d1c-11cf-861e-0020af6e7c57 v0.0: IRemoteActivation (IActivation)		
	0x00	RemoteActivation

Starting with Windows 2000, the **ISystemActivator** COM interface is used instead of the **IRemoteActivation** RPC interface.

The IOXIDResolver RPC interface (formerly known as IObjectExporter) is remotely used to reach the local object resolver (OR). The Object Resolver component is in charge to:

- return protocol sequences, string bindings and machine id for an object server, given its OXID (**ResolveOXID()** and **ResolveOXID2()** (only supported by DCOM version 5.2 and above))
- respond to ping requests (**SimplePing()** and **ComplexPing()** functions)
- respond to **ServerAlive()** and **ServerAlive2()** functions requests

The interface identifier of IOXIDResolver is:

```
99fcfec4-5260-101b-bbcb-00aa0021347a v0.0: IOXIDResolver
```

Table 4.8. IOXIDResolver operations

Interface	Operation number	Operation name
99fcfec4-5260-101b-bbcb-00aa0021347a v0.0: IOXIDResolver		
	0x00	ResolveOxid
	0x01	SimplePing
	0x02	ComplexPing
	0x03	ServerAlive
	0x04	ResolveOxid2
	0x05	ServerAlive2

There is also a local version of the IOXIDResolver:

```
e60c73e6-88f9-11cf-9af1-0020af6e72f4 v2.0: ILocalObjectExporter
```

Table 4.9. ILocalObjectExporter operations

Interface	Operation number	Operation name
e60c73e6-88f9-11cf-9af1-0020af6e72f4 v2.0: ILocalObjectExporter		
	0x00	Connect
	0x01	AllocateReservedIds
	0x02	BulkUpdateOIDs
	0x03	ClientResolveOXID
	0x04	ServerAllocateOXIDandOIDs
	0x05	ServerAllocateOIDs
	0x06	ServerFreeOXIDAndOIDs
	0x07	Disconnect

For more information about the DCOM transport into DCE RPC, see [73].

The ISCM RPC interface is a local interface used by local applications to communicate with the local COM SCM:

```
412f241e-c12a-11ce-abff-0020af6e7a17 v0.2: ISCM
```

Table 4.10. ISCM operations

Interface	Operation number	Operation name
412f241e-c12a-11ce-abff-0020af6e7a17 v0.2: ISCM		
	0x00	ServerRegisterClsid
	0x01	ServerRevokeClsid
	0x02	GetThreadID
	0x03	UpdateActivationSettings
	0x04	RegisterWindowPropInterface
	0x05	GetWindowPropInterface
	0x06	EnableDisableDynamicIPTracking
	0x07	GetCurrentAddrExclusionList
	0x08	SetAddrExclusionList
	0x09	FlushSCMBindings
	0x0a	RetireServer

The IROT RPC interface is used by local processes to access the Running Object Table (ROT), to register or unregister COM objects:

```
b9e79e60-3d52-11ce-aaa1-00006901293f v0.2:IROT
```

Table 4.11. IROT operations

Interface	Operation number	Operation name
b9e79e60-3d52-11ce-aaa1-00006901293f v0.2: IROT		
	0x00	IrotRegister
	0x01	IRotRevoke
	0x02	IrotIsRunning
	0x03	IrotGetObject
	0x04	IrotNoteChangeTime
	0x05	IrotGetTimeOfLastChange
	0x06	IrotEnumRunning

The IMachineActivatorControl is also a local interface used to notify the COM SCM when COM surrogates start or stop:

```
c6f3ee72-ce7e-11d1-b71e-00c04fc3111a v1.0: IMachineActivatorControl
```

Table 4.12. IMachineActivatorControl operations

Interface	Operation number	Operation name
c6f3ee72-ce7e-11d1-b71e-00c04fc3111a v1.0: IMachineActivatorControl		
	0x00	ProcessActivatorStarted
	0x01	ProcessActivatorInitializing
	0x02	ProcessActivatorReady
	0x03	ProcessActivatorStopped
	0x04	ProcessActivatorPaused
	0x05	ProcessActivatorResumed
	0x06	ProcessActivatorUserInitializing

Prev | Up | Next
4.8.2. RPC interfaces supported by the rpcss service | Home | 4.8.4. ORPC interfaces running in the rpcss service

4.8.4. ORPC interfaces running in the rpcss service

ORPC (Object RPC) services are used by DCOM (Distributed COM). ORPC calls can be distinguished from RPC calls because, on the wire, they always have an implicit parameter, either of type ORPCTHIS or ORPCTHAT (see section 3.2 of Section 4.18, “DCOM”).

Also, versions of ORPC services interface identifiers is always 0.0, as explained in [73] :

> Finally, the interface version number (named if_vers) must always be 0.0. This is because a COM interface may never be modified after it is published. COM interfaces are not versioned; a new interface is defined instead.

The following ORPC services are running in the rpcss service:

```
00000136-0000-0000-c000-000000000046 v0.0: ISCMActivator
000001a0-0000-0000-c000-000000000046 v0.0: ISystemActivator
```

ISCMActivator is an ORPC interface implemented by the COM SCM to handle remote activation requests (**CoCreateInstance()**, **CoGetClassObject()** ...) :

Table 4.13. ISCMActivator operations

Interface	Operation number	Operation name
00000136-0000-0000-c000-000000000046 v0.0: ISCMActivator		
	0x00	QueryInterfaceSCMActivator
	0x01	AddRefISCMActivator
	0x02	ReleaseISCMActivator
	0x03	SCMActivatorGetClassObject
	0x04	SCMActivatorCreateInstance

ISystemActivator (also known as IRemoteSCMActivator) is an ORPC base interface that must be implemented by servers supporting object activation. In the specific case of the COM SCM, running inside the rpcss service, this interface is used when the activation process is looking for an object, asking

to the local or a remote SCM to activate a given object.

Table 4.14. ISystemActivator (IRemoteSCMActivator) operations

Interface	Operation number	Operation name
000001a0-0000-0000-c000-000000000046 v0.0: ISystemActivator (IRemoteSCMActivator)		
	0x00	QueryInterfaceIRemoteSCMActivator
	0x01	AddRefIRemoteISCMActivator
	0x02	ReleaseIRemoteISCMActivator
	0x03	RemoteGetClassObject
	0x04	RemoteCreateInstance

Prev | Up | Next
4.8.3. DCOM-related RPC interfaces running in the rpcss service | Home | 4.9. Windows core MSRPC interfaces

4.9. Windows core MSRPC interfaces

4.9.1. lsarpc interface
4.9.2. samr interface
4.9.3. netlogon interface
4.9.4. drsuapi interface
4.9.5. dssetup interface
4.9.6. eventlog interface
4.9.7. pnp interface
4.9.8. srvsvc interface
4.9.9. svcctl interface
4.9.10. winreg interface
4.9.11. wkssvc interface

Windows NT and Active Directory domains are built on the following MSRPC interfaces:

- Section 4.9.1, “lsarpc interface”: Local Security Authority (LSA) RPC service
- Section 4.9.2, “samr interface”: Security Account Manager RPC service
- Section 4.9.3, “netlogon interface”: Netlogon RPC service
- Section 4.9.4, “drsuapi interface”: Active Directory replication service
- Section 4.9.5, “dssetup interface”: Active Directory setup

Remote administration of Windows services is implemented using the following MSRPC interfaces:

- Section 4.9.6, “eventlog interface”: Eventlog service management
- Section 4.9.7, “pnp interface”: Plug and Play service management
- Section 4.9.8, “srvsvc interface”: Server service management
- Section 4.9.9, “svcctl interface”: Windows services management
- Section 4.9.10, “winreg interface”: Windows remote registry access
- Section 4.9.11, “wkssvc interface”: Workstation service management

Windows Vista introduced the following MSRPC core interfaces:

- Section 4.12.8, “Windows event logging service”: Windows event logging service management

---

4.9.1. lsarpc interface

The **lsarpc** interface is used to communicate with the LSA (Local Security Authority) subsystem.

Before Windows 2000, the **lsarpc** interface is only available on the **lsarpc** named pipe endpoint:

```
C:\> ifids -p ncacn_np -e \pipe\lsarpc \\.

Interfaces: 4
  12345778-1234-abcd-ef00-0123456789ab v0.0

[...]
```

In Active Directory domains (and particularly, Active Directory domain controllers) and Windows Server 2003 systems, the **lsarpc** interface is also available (and used) over a TCP endpoint:

```
C:\> ifids -p ncacn_ip_tcp -e 1025 127.0.0.1

Interfaces: 12
  12345778-1234-abcd-ef00-0123456789ab v0.0

[...]
```

Starting with Windows Server 2003 SP1, some operations of the lsarpc interface can only be used over a specific protocol sequence.

IDL (Interface Definition Language) for the **lsarpc** interface is available in Samba 4 [53].

Table 4.15. lsarpc operations

Interface	Operation number	Operation name	Windows API

12345778-1234-abcd-ef00-0123456789ab v0.0: lsarpc			
	0x00	LsarClose	LsaClose
	0x01	LsarDelete	
	0x02	LsarEnumeratePrivileges	
	0x03	LsarQuerySecurityObject	
	0x04	LsarSetSecurityObject	
	0x05	LsarChangePassword	
	0x06	LsarOpenPolicy	LsaOpenPolicy
	0x07	LsarQueryInformationPolicy	LsaQueryInformationPolicy
	0x08	LsarSetInformationPolicy	LsaSetInformationPolicy
	0x09	LsarClearAuditLog	
	0x0a	LsarCreateAccount	
	0x0b	LsarEnumerateAccounts	
	0x0c	LsarCreateTrustedDomain	
	0x0d	LsarEnumerateTrustedDomains	
	0x0e	LsarLookupNames	LsaLookupNames
	0x0f	LsarLookupSids	LsaLookupSids
	0x10	LsarCreateSecret	
	0x11	LsarOpenAccount	
	0x12	LsarEnumeratePrivilegesAccount	LsaEnumerateAccountRights
	0x13	LsarAddPrivilegesToAccount	LsaAddAccountRights
	0x14	LsarRemovePrivilegesFromAccount	LsaRemoveAccountRights
	0x15	LsarGetQuotasForAccount	
	0x16	LsarSetQuotasForAccount	
	0x17	LsarGetSystemAccessAccount	
	0x18	LsarSetSystemAccessAccount	
	0x19	LsarOpenTrustedDomain	
	0x1a	LsarQueryInfoTrustedDomain	
	0x1b	LsarSetInformationTrustedDomain	

	0x1c	LsarOpenSecret	
	0x1d	LsarSetSecret	
	0x1e	LsarQuerySecret	
	0x1f	LsarLookupPrivilegeValue	
	0x20	LsarLookupPrivilegeName	
	0x21	LsarLookupPrivilegeDisplayName	
	0x22	LsarDeleteObject	
	0x23	LsarEnumerateAccountsWithUserRight	
	0x24	LsarEnumerateAccountRights	
	0x25	LsarAddAccountRights	
	0x26	LsarRemoveAccountRights	
	0x27	LsarQueryTrustedDomainInfo	LsaQueryTrustedDomainInfo
	0x28	LsarSetTrustedDomainInfo	LsaSetTrustedDomainInformation
> Windows 2000	0x29	LsarDeleteTrustedDomain	LsaDeleteTrustedDomain
-	0x2a	LsarStorePrivateData	LsaStorePrivateData
-	0x2b	LsarRetrievePrivateData	LsaRetrievePrivateData
-	0x2c	LsarOpenPolicy2	LsaOpenPolicy
-	0x2d	LsarGetUserName	
-	0x2e	LsarQueryInformationPolicy2	
-	0x2f	LsarSetInformationPolicy2	
-	0x30	LsarQueryTrustedDomainInfoByName	
-	0x31	LsarSetTrustedDomainInfoByName	
-	0x32	LsarEnumerateTrustedDomainsEx	
-	0x33	LsarCreateTrustedDomainEx	
-	0x34	LsarCloseTrustedDomainEx	
-	0x35	LsarQueryDomainInformationPolicy	LsaQueryDomainInformationPolicy
-	0x36	LsarSetDomainInformationPolicy	LsaSetDomainInformationPolicy
-	0x37	LsarOpenTrustedDomainByName	
-	0x38	LsarTestCall	
-	0x39	LsarLookupSids2	LsaLookupSids
-	0x3a	LsarLookupNames2	LsaLookupNames2

-	0x3b	LsarCreateTrustedDomainEx2	
> Windows 2000 Service Pack 3 (SP3)	0x3c	CredrWrite	CredWrite
-	0x3d	CredrRead	CredRead
-	0x3e	CredrEnumerate	CredEnumerate
-	0x3f	CredrWriteDomainCredentials	CredWriteDomainCredentials
-	0x40	CredrReadDomainCredentials	CredReadDomainCredentials
-	0x41	CredrDelete	CredDelete
-	0x42	CredrGetTargetInfo	CredGetTargetInfo
-	0x43	CredrProfileLoaded	
-	0x44	LsarLookupNames3	
-	0x45	CredrGetSessionTypes	CredGetSessionTypes
-	0x46	LsarRegisterAuditEvent	
-	0x47	LsarGenAuditEvent	
-	0x48	LsarUnregisterAuditEvent	
-	0x49	LsarQueryForestTrustInformation	
-	0x4a	LsarSetForestTrustInformation	
-	0x4b	CredrRename	CredRename
-	0x4c	LsarLookupSids3	
-	0x4d	LsarLookupNames4	
-	0x4e	LsarOpenPolicySce	
> Windows Server 2003	0x4f	LsarAdtRegisterSecurityEventSource	
-	0x50	LsarAdtUnregisterSecurityEventSource	
-	0x51	LsarAdtReportSecurityEvent	
> Windows Vista	0x52	CredrFindBestCredential	
-	0x53	LsarSetAuditPolicy	
-	0x54	LsarQueryAuditPolicy	
-	0x55	LsarEnumerateAuditPolicy	
-	0x56	LsarEnumerateAuditCategories	
-	0x57	LsarEnumerateAuditSubCategories	

-	0x58	LsarLookupAuditCategoryName		
-	0x59	LsarLookupAuditSubCategoryName		
-	0x5a	LsarSetAuditSecurity		
-	0x5b	LsarQueryAuditSecurity		
-	0x5c	CredReadByTokenHandle		
-	0x5d	CredrRestoreCredentials		
-	0x5e	CredrBackupCredentials		

To obtain a handle to the LSA rpc server, one of the following operations must be used:

- LsarOpenPolicy (0x06)
- LsarOpenPolicy2 (0x2c)

Opened handle are supposed to be closed with the following operation:

- LsarClose (0x00)

To resolve SID to names and vice-versa, the following operations are supported:

- LsarLookupNames (0x0e)
- LsarLookupSids (0x0f)
- LsarLookupNames2 (0x3a)
- LsarLookupSids2 (0x39)
- LsarLookupNames3 (0x44)
- LsarLookupSids3 (0x4c)
- LsarLookupNames4 (0x4d)

To obtain system names (Se*) of security privileges supported by the LSA, the following operation can be used:

- LsarEnumeratePrivileges (0x02)

To convert between privileges system names, numeric values and descriptions, the following operations can be used:

- LsarLookupPrivilegeValue (0x1f)
- LsarLookupPrivilegeName (0x20)
- LsarLookupPrivilegeDisplayName (0x21)

To query or set parameters of the LSA policy, the following operation are supported:

- LsarQueryInformationPolicy (0x07)
- LsarSetInformationPolicy (0x08)
- LsarQueryInformationPolicy2 (0x2e)
- LsarSetInformationPolicy2 (0x2f)

To open an account, given its SID, the following operation is used:

- LsarOpenAccount (0x11)

The following operations can be used with an opened handle returned by the LsarOpenAccount operation:

- LsarEnumeratePrivilegesAccount (0x12)
- LsarAddPrivilegesToAccount (0x13)
- LsarRemovePrivilegesFromAccount (0x14)
- LsarGetQuotasForAccount (0x15)
- LsarSetQuotasForAccount (0x16)
- LsarGetSystemAccessAccount (0x17)
- LsarSetSystemAccessAccount (0x18)
- LsarEnumerateAccountRights (0x24)
- LsarAddAccountRights (0x25)
- LsarRemoveAccountRights (0x26)

To manage trusted domains, the following operations are available:

- LsarEnumerateTrustedDomains (0x0d)
- LsarEnumerateTrustedDomainsEx (0x32)
- LsarCreateTrustedDomain (0x0c)
- LsarCreateTrustedDomainEx (0x33)
- LsarCreateTrustedDomainEx2 (0x3b)
- LsarOpenTrustedDomain (0x19)
- LsarOpenTrustedDomainByName (0x37)
- LsarQueryTrustedDomainInfo (0x27)
- LsarQueryTrustedDomainInfoByName (0x30)
- LsarSetTrustedDomainInfo (0x28)
- LsarSetTrustedDomainInfoByName (0x31)
- LsarCloseTrustedDomainEx (0x34)
- LsarDeleteTrustedDomain (0x29)

To manipulate LSA secrets, the following operations are available:

- LsarCreateSecret (0x10)
- LsarOpenSecret (0x1c)
- LsarSetSecret (0x1d)
- LsarQuerySecret (0x1e)

- LsarDelete (0x01)

To get and set ACL on LSA objects, the following operations are available:

- LsarQuerySecurityObject (0x03)
- LsarSetSecurityObject (0x04)

4.9.2. samr interface

The **samr** interface is used to communicate with the SAM (Security Account Manager) subsystem.

Before Windows 2000, the **samr** interface is only available on the **samr** named pipe endpoint:

```
C:\> ifids -p ncacn_np -e \pipe\samr \\.

Interfaces: 4
[...]

  12345778-1234-abcd-ef00-0123456789ac v0.0

[...]
```

In Active Directory domains (and particularly, Active Directory domain controllers), the **samr** interface is also available (and used) over a TCP endpoint:

```
C:\> ifids -p ncacn_ip_tcp -e 1025 127.0.0.1

Interfaces: 12
[...]

  12345778-1234-abcd-ef00-0123456789ac v0.0

[...]
```

During Active Directory domain joins, the creation of computer accounts is implemented with **samr** operations called on the TCP endpoint of Active Directory domain controllers.

IDL (Interface Definition Language) for the **samr** interface is available in Samba 4 [55].

Table 4.16. samr operations

Interface	Operation number	Operation name
12345778-1234-abcd-ef00-0123456789ac v1.0: samr		
	0x00	SamrConnect
	0x01	SamrCloseHandle
	0x02	SamrSetSecurityObject
	0x03	SamrQuerySecurityObject
	0x04	SamrShutdownSamServer
	0x05	SamrLookupDomainInSamServer
	0x06	SamrEnumerateDomainsInSamServer
	0x07	SamrOpenDomain
	0x08	SamrQueryInformationDomain
	0x09	SamrSetInformationDomain
	0x0a	SamrCreateGroupInDomain
	0x0b	SamrEnumerateGroupsInDomain
	0x0c	SamrCreateUserInDomain
	0x0d	SamrEnumerateUsersInDomain
	0x0e	SamrCreateAliasInDomain
	0x0f	SamrEnumerateAliasesInDomain
	0x10	SamrGetAliasMembership
	0x11	SamrLookupNamesInDomain
	0x12	SamrLookupIdsInDomain
	0x13	SamrOpenGroup
	0x14	SamrQueryInformationGroup
	0x15	SamrSetInformationGroup
	0x16	SamrAddMemberToGroup
	0x17	SamrDeleteGroup
	0x18	SamrRemoveMemberFromGroup
	0x19	SamrGetMembersInGroup
	0x1a	SamrSetMemberAttributesOfGroup
	0x1b	SamrOpenAlias
	0x1c	SamrQueryInformationAlias

	0x1d	SamrSetInformationAlias
	0x1e	SamrDeleteAlias
	0x1f	SamrAddMemberToAlias
	0x20	SamrRemoveMemberFromAlias
	0x21	SamrGetMembersInAlias
	0x22	SamrOpenUser
	0x23	SamrDeleteUser
	0x24	SamrQueryInformationUser
	0x25	SamrSetInformationUser
	0x26	SamrChangePasswordUser
	0x27	SamrGetGroupsForUser
	0x28	SamrQueryDisplayInformation
	0x29	SamrGetDisplayEnumerationIndex
	0x2a	SamrTestPrivateFunctionsDomain
	0x2b	SamrTestPrivateFunctionsUser
	0x2c	SamrGetUserDomainPasswordInformation
> Windows 2000	0x2d	SamrRemoveMemberFromForeignDomain
-	0x2e	SamrQueryInformationDomain2
-	0x2f	SamrQueryInformationUser2
-	0x30	SamrQueryDisplayInformation2
-	0x31	SamrGetDisplayEnumerationIndex2
-	0x32	SamrCreateUser2InDomain
-	0x33	SamrQueryDisplayInformation3
-	0x34	SamrAddMultipleMembersToAlias
-	0x35	SamrRemoveMultipleMembersFromAlias
-	0x36	SamrOemChangePasswordUser2
-	0x37	SamrUnicodeChangePasswordUser2
-	0x38	SamrGetDomainPasswordInformation
-	0x39	SamrConnect2
-	0x3a	SamrSetInformationUser2
-	0x3b	SamrSetBootKeyInformation
-	0x3c	SamrGetBootKeyInformation

-	0x3d	SamrConnect3
-	0x3e	SamrConnect4
-	0x3f	SamrUnicodeChangePasswordUser3
> Windows XP and Windows Server 2003	0x40	SamrConnect5
-	0x41	SamrRidToSid
-	0x42	SamrSetDSRMPassword
-	0x43	SamrValidatePassword
> Windows Vista	0x44	SamrQueryLocalizableAccountsInDomain
-	0x45	SamrPerformGenericOperation

To connect to the SAM server, one of the following operations are used:

- SamrConnect (0x00)
- SamrConnect2 (0x39)
- SamrConnect3 (0x3d)
- SamrConnect4 (0x3e)
- SamrConnect5 (0x40)

Then, available domains in the SAM server can be enumerated using the following operation:

- SamrEnumerateDomainsInSamServer (0x06)

The following operation is used to obtain the SID of a domain, given its name:

- SamrLookupDomainInSamServer (0x05)

This operation typically returns the BUILTIN domain (S-1-5-32) and the machine domain (local domain for a non-domain controller machine, NT 4 or Active Directory domain for a domain controller machine).

The domain SID can then be used to open a given domain:

- SamrOpenDomain (0x07)

General information about the opened domain can be obtained or set with the following operations:

- SamrQueryInformationDomain (0x08)

- SamrQueryInformationDomain2 (0x2e)
- SamrSetInformationDomain (0x09)

Once a domain is opened, it is possible to enumerate groups, aliases and users, using the following operations:

- SamrEnumerateGroupsInDomain (0x0b)
- SamrEnumerateAliasesInDomain (0x0f)
- SamrEnumerateUsersInDomain (0x0d)

RID and names resolution inside an opened domain are implemented by the following operations:

- SamrLookupNamesInDomain (0x11)
- SamrLookupIdsInDomain (0x12)

Domain password policies can be obtained with the following operations:

- SamrGetUserDomainPasswordInformation (0x2c)
- SamrGetDomainPasswordInformation (0x38)

To create a new group, alias or user in the opened domain, the following operations can be used:

- SamrCreateGroupInDomain (0x0a)
- SamrCreateAliasInDomain (0x0e)
- SamrCreateUserInDomain (0x0c)
- SamrCreateUser2InDomain (0x32)

To open an existing group, alias or user in the opened domain, the following operations exist:

- SamrOpenGroup (0x13)
- SamrOpenAlias (0x1b)
- SamrOpenUser (0x22)

To delete an existing group, alias or user in the opened domain, the following operations exist:

- SamrDeleteGroup (0x17)
- SamrDeleteAlias (0x1e)
- SamrDeleteUser (0x23)

To obtain a list of members in groups or aliases, the following operations can be used:

- SamrGetMembersInGroup (0x19)
- SamrGetMembersInAlias (0x21)

To add or remove a member to a group or alias, the following operations are available:

- SamrAddMemberToGroup (0x16)
- SamrAddMemberToAlias (0x1f)
- SamrRemoveMemberFromGroup (0x18)
- SamrRemoveMemberFromAlias (0x20)

For aliases, it is also possible to add or remove multiple members to or from an alias:

- SamrAddMultipleMembersToAlias (0x34)
- SamrRemoveMultipleMembersFromAlias (0x35)

To obtain or set information about a given group or alias, the following operations exist:

- SamrQueryInformationGroup (0x14)
- SamrQueryInformationAlias (0x1c)
- SamrSetInformationGroup (0x15)
- SamrSetInformationAlias (0x1d)

Similar operations exist for accounts management:

- SamrQueryInformationUser (0x24)
- SamrQueryInformationUser2 (0x2f)
- SamrSetInformationUser (0x25)
- SamrSetInformationUser2 (0x3a)

A list of groups containing a given user can be obtained with the following operation:

- SamrGetGroupsForUser (0x27)

Finally, handles returned by the following operations are supposed to be closed, using the SamrCloseHandle (0x01) operation:

- SamrConnect (0x00)
- SamrConnect2 (0x39)
- SamrConnect3 (0x3d)
- SamrConnect4 (0x3e)
- SamrConnect5 (0x40)
- SamrOpenDomain (0x07)

- SamrOpenGroup (0x13)
- SamrOpenAlias (0x1b)
- SamrOpenUser (0x22)
- SamrCreateUserInDomain (0x0c)
- SamrCreateUser2InDomain (0x32)
- SamrCreateAliasInDomain (0x0e)
- SamrCreateGroupInDomain (0x0a)

4.9.3. netlogon interface

The **netlogon** interface is used to communicate with the netlogon service, that typically run on member servers and domain controllers.

IDL (Interface Definition Language) for the **netlogon** interface is available in Samba 4 [56].

Table 4.17. netlogon operations

Interface	Operation number	Operation name	Windows API
12345678-1234-abcd-ef00-01234567cffb v1.0: netlogon			
	0x00	NetrLogonUasLogon	
	0x01	NetrLogonUasLogoff	
	0x02	NetrLogonSamLogon	
	0x03	NetrLogonSamLogoff	
	0x04	NetrServerReqChallenge	
	0x05	NetrServerAuthenticate	
	0x06	NetrServerPasswordSet	
	0x07	NetrDatabaseDeltas	
	0x08	NetrDatabaseSync	
	0x09	NetrAccountDeltas	
	0x0a	NetrAccountSync	
	0x0b	NetrGetDCName	NetGetDCName
	0x0c	NetrLogonControl	
	0x0d	NetrGetAnyDCName	NetGetAnyDCName
	0x0e	NetrLogonControl2	

	0x0f	NetrServerAuthenticate2	
	0x10	NetrDatabaseSync2	
	0x11	NetrDatabaseRedo	
	0x12	NetrLogonControl2Ex	
	0x13	NetrEnumerateTrustedDomains	
> Windows 2000	0x14	DsrGetDcName	DsGetDCName
-	0x15	NetrLogonDummyRoutine1	
-	0x16	NetrLogonSetServiceBits	
-	0x17	NetrLogonGetTrustRid	
-	0x18	NetrLogonComputeServerDigest	
-	0x19	NetrLogonComputeClientDigest	
-	0x1a	NetrServerAuthenticate3	
-	0x1b	DsrGetDcNameEx	DsGetDCName
-	0x1c	DsrGetSiteName	DsGetSiteName
-	0x1d	NetrLogonGetDomainInfo	
-	0x1e	NetrServerPasswordSet2	
-	0x1f	NetrServerPasswordGet	
-	0x20	NetrLogonSendToSam	
-	0x21	DsrAddressToSiteNamesW	DsAddressToSiteNames
-	0x22	DsrGetDcNameEx2	DsGetDCName
-	0x23	NetrLogonGetTimeServiceParentDomain	
-	0x24	NetrEnumerateTrustedDomainsEx	
-	0x25	DsrAddressToSiteNamesExW	DsAddressToSiteNames
-	0x26	DsrGetDcSiteCoverageW	DsGetDcSiteCoverage
-	0x27	NetrLogonSamLogonEx	
-	0x28	DsrEnumerateDomainTrusts	DsEnumerateDomainTrusts
-	0x29	DsrDeregisterDnsHostRecords	DsDeregisterDnsHostRecords
-	0x2a	NetrServerTrustPasswordsGet	

> Windows XP and Windows Server 2003	0x2b	DsrGetForestTrustInformation	DsGetForestTrustInformationW
-	0x2c	NetrGetForestTrustInformation	
-	0x2d	NetrLogonSamLogonWithFlags	
-	0x2e	NetrServerGetTrustInfo	

4.9.4. drsuapi interface

The **drsuapi** interface is used between Active Directory domain controllers for replication:

```
Active Directory replication interface: e3514235-4b06-11d1-ab04-00c04fc2dcd2 v4.0
```

IDL (Interface Definition Language) for the **drsuapi** interface is available in Samba 4 [80].

It supports the following operations:

Table 4.18. drsuapi operations

Interface	Operation number	Operation name	Windows API
e3514235-4b06-11d1-ab04-00c04fc2dcd2 v4.0: drsuapi			
	0x00	DRSBind	DsBind
	0x01	DRSUnbind	DsUnBind
	0x02	DRSReplicaSync	DsReplicaSync
	0x03	DRSGetNCChanges	
	0x04	DRSUpdateRefs	
	0x05	DRSReplicaAdd	DsReplicaAdd
	0x06	DRSReplicaDel	DsReplicaDel
	0x07	DRSReplicaModify	DsReplicaModify
	0x08	DRSVerifyNames	
	0x09	DRSGetMemberships	
	0x0a	DRSInterDomainMove	
	0x0b	DRSGetNT4ChangeLog	
	0x0c	DRSCrackNames	DsCrackNames
	0x0d	DRSWriteSPN	DsWriteAccountSpn
	0x0e	DRSRemoveDsServer	
	0x0f	DRSRemoveDsDomain	
	0x10	DRSDomainControllerInfo	
	0x11	DRSAddEntry	
	0x12	DRSExecuteKCC	
	0x13	DRSGetReplInfo	

	0x14	DRSAddSidHistory	DsAddSidHistory
> Windows Server 2003 and >	0x15	DRSGetMemberships2	
-	0x16	DRSReplicaVerifyObjects	
-	0x17	DRSGetObjectExistence	
-	0x18	DRSQuerySitesByCost	DsQuerySitesByCost

Ethereal has a dissector for this interface [83]. It is particularly useful when used with the Kerberos decryption feature: in that case, encrypted operations are dissected.

4.9.5. dssetup interface

The **dssetup** interface (Directory Services Setup) is used in Active Directory environments. The first operation, **DsRolerGetPrimaryDomainInformation**, is used to query the configuration of an Active Directory domain member system.

IDL (Interface Definition Language) for the **dssetup** interface is available in Samba 4 [54].

The **dssetup** interface runs in the LSA on Windows 2000 and later and supports at least one operation:

Table 4.19. dssetup operations

Interface	Operation number	Operation name	Windows API
3919286a-b10c-11d0-9ba8-00c04fd92ef5 v0.0: dssetup			
Windows 2000 and >	0x00	DsRolerGetPrimaryDomainInformation	DsRoleGetPrimaryDomainInformation
Windows 2000 and Windows XP (before MS04-011)	0x01	DsRolerDnsNameToFlatName	
-	0x02	DsRolerDcAsDc	
-	0x03	DsRolerDcAsReplica	
-	0x04	DsRolerDemoteDc	
-	0x05	DsRolerGetDcOperationProgress	
-	0x06	DsRolerGetDcOperationResults	
-	0x07	DsRolerCancel	
-	0x08	DsRolerServerSaveStateForUpgrade	
-	0x09	DsRolerUpgradeDownlevelServer	
-	0x0a	DsRolerAbortDownlevelServerUpgrade	

A buffer overflow in a logging function in **lsasrv.dll** was discovered by eEye [84] on 2004/04/13 and fixed in the MS04-011 [85] Microsoft security patch. This buffer overflow can be specifically exploited with the **DsRolerUpgradeDownlevelServer** operation to gain the SYSTEM privilege, because this specific operation does not impersonate the security context of the caller (i.e., does not call **RpcImpersonateClient()**).

This buffer overflow has been exploited by the Sasser worm [86], discovered on 2004/04/30.

Starting with Windows Server 2003, these operations belong to the **dsrole** interface, which can not be accessed remotely, as explained below. Only the first operation, **DsRolerGetPrimaryDomainInformation**, is available in the **dssetup** interface.

The MS04-011 security patch also removed all operations of the **dssetup** interface except the first one (**DsRolerGetPrimaryDomainInformation**) on Windows 2000 and Windows XP.

Prev	Up	Next
4.9.4. drsuapi interface	Home	4.9.6. eventlog interface

4.9.6. eventlog interface

The **eventlog** interface can be used to access to Windows NT eventlogs.

IDL (Interface Definition Language) for the **eventlog** interface is available in Samba 4 [57].

Table 4.20. eventlog operations

Interface	Operation number	Operation name	Windows API
82273fdc-e32a-18c3-3f78-827929dc23ea v0.0: eventlog			
	0x00	ElfrClearELFW	ClearEventLog
	0x01	ElfrBackupELFW	BackupEventLog
	0x02	ElfrCloseEL	CloseEventLog
	0x03	ElfrDeregisterEventSource	DeregisterEventSource
	0x04	ElfrNumberOfRecords	GetNumberOfEventLogRecords
	0x05	ElfrOldestRecord	GetOldestEventLogRecord
	0x06	ElfrChangeNotify	NofifyChangeEventLog
	0x07	ElfrOpenELW	OpenEventLog
	0x08	ElfrRegisterEventSourceW	RegisterEventSource
	0x09	ElfrOpenBELW	OpenBackupEventlog
	0x0a	ElfrReadELW	ReadEventLog
	0x0b	ElfrReportEventW	ReportEvent
	0x0c	ElfrClearELFA	ClearEventLog
	0x0d	ElfrBackupELFA	BackupEventLog
	0x0e	ElfrOpenELA	OpenEventLog

	0x0f	ElfrRegisterEventSourceA	RegisterEventSource
	0x10	ElfrOpenBELA	OpenBackupEventlog
> Windows 2000	0x11	ElfrReadELA	ReadEventLog
-	0x12	ElfrReportEventA	ReportEvent
-	0x13	ElfrRegisterClusterSvc	
-	0x14	ElfrDeregisterClusterSvc	
-	0x15	ElfrWriteClusterEvents	
-	0x16	ElfrGetLogInformation	GetEventLogInformation
> Windows XP	0x17	ElfrFlushEL	
> Windows Server 2003	0x18	ElfrReportEventAndSourceW	

Operations in the eventlog interface that take Unicode strings as parameters end with W and operations that take ASCII strings as parameters end with A.

Opening an eventlog:

- ElfrOpenELW (0x07)
- ElfrOpenELA (0x0e)

Obtaining general information about an opened eventlog:

- ElfrGetLogInformation (0x16)

Opening the backup of an eventlog:

- ElfrOpenBELW (0x09)
- ElfrOpenBELA (0x10)

Obtaining the number of records in an opened eventlog:

- ElfrNumberOfRecords (0x04)

Obtaining the oldest record number in an opened eventlog:

- ElfrOldestRecord (0x05)

Reading records stored in an opened eventlog, the following operations are used:

- ElfrReadELW (0x0a)
- ElfrReadELA (0x11)

Backing up an opened eventlog:

- ElfrBackupELFW (0x01)
- ElfrBackupELFA (0x0d)

Clearing the content of an opened eventlog:

- ElfrClearELFW (0x00)
- ElfrClearELFA (0x0c)

Registering an event source (in the registry):

- ElfrRegisterEventSourceW (0x08)
- ElfrRegisterEventSourceA (0x0f)

Reporting an event in an opened eventlog:

- ElfrReportEventW (0x0b)
- ElfrReportEventA (0x12)

Flushing an opened eventlog:

- ElfrFlushEL (0x17)

Closing an opened eventlog:

- ElfrCloseEL (0x02)

4.9.7. pnp interface

The Plug and Play service runs one RPC service, **pnp**:

```
Z:\>ifids -p ncalrpc -e ntsvcs serveur
Interfaces: 7

[...]

  8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0

Z:\>ifids -p ncacn_np -e \pipe\ntsvcs \\.
Interfaces: 7

[...]

  8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0
```

Before Windows Vista, the **\pipe\ntsvcs** named pipe endpoint is usually used to reach the pnp interface.

In Windows Vista, a dedicated named pipe, **plugplay** was introduced.

```
C:\Users\>ifids -p ncacn_np -e \pipe\plugplay \\.
Interfaces: 3
  8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0

  [...]
```

Table 4.21. pnp operations

Interface	Operation number	Operation name
8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0: pnp		

	0x00	PNP_Disconnect
	0x01	PNP_Connect
	0x02	PNP_GetVersion
	0x03	PNP_GetGlobalState
	0x04	PNP_InitDetection
	0x05	PNP_ReportLogOn
	0x06	PNP_ValidateDeviceInstance
	0x07	PNP_GetRootDeviceInstance
	0x08	PNP_GetRelatedDeviceInstance
	0x09	PNP_EnumerateSubKeys
	0x0a	PNP_GetDeviceList
	0x0b	PNP_GetDeviceListSize
	0x0c	PNP_GetDepth
	0x0d	PNP_GetDeviceRegProp
	0x0e	PNP_SetDeviceRegProp
	0x0f	PNP_GetClassInstance
	0x10	PNP_CreateKey
	0x11	PNP_DeleteRegistryKey
	0x12	PNP_GetClassCount
	0x13	PNP_GetClassName
	0x14	PNP_DeleteClassKey
	0x15	PNP_GetInterfaceDeviceAlias
	0x16	PNP_GetInterfaceDeviceList
	0x17	PNP_GetInterfaceDeviceListSize
	0x18	PNP_RegisterDeviceClassAssociation
	0x19	PNP_UnregisterDeviceClassAssociation
	0x1a	PNP_GetClassRegProp
	0x1b	PNP_SetClassRegProp
	0x1c	PNP_CreateDevInst
	0x1d	PNP_DeviceInstanceAction
	0x1e	PNP_GetDeviceStatus
	0x1f	PNP_SetDeviceProblem

	0x20	PNP_DisableDevInst
	0x21	PNP_UninstallDevInst
	0x22	PNP_AddID
	0x23	PNP_RegisterDriver
	0x24	PNP_QueryRemove
	0x25	PNP_RequestDeviceEject
	0x26	PNP_IsDockStationPresent
	0x27	PNP_RequestEjectPC
	0x28	PNP_HwProfFlags
	0x29	PNP_GetHwProfInfo
	0x2a	PNP_AddEmptyLogConf
	0x2b	PNP_FreeLogConf
	0x2c	PNP_GetFirstLogConf
	0x2d	PNP_GetNextLogConf
	0x2e	PNP_GetLogConfPriority
	0x2f	PNP_AddResDes
	0x30	PNP_FreeResDes
	0x31	PNP_GetNextResDes
	0x32	PNP_GetResDesData
	0x33	PNP_GetResDesDataSize
	0x34	PNP_ModifyResDes
	0x35	PNP_DetectResourceConflict
	0x36	PNP_QueryResConfList
	0x37	PNP_SetHwProf
	0x38	PNP_QueryArbitratorFreeData
	0x39	PNP_QueryArbitratorFreeSize
	0x3a	PNP_RunDetection
	0x3b	PNP_RegisterNotification
	0x3c	PNP_UnregisterNotification
> Windows XP and Windows Server 2003	0x3d	PNP_GetCustomDevProp
	0x3e	PNP_GetVersionInternal

	0x3f	PNP_GetBlockedDriverInfo
	0x40	PNP_GetServerSideDeviceInstallFlags
> Windows Vista	0x41	PNP_GetObjectPropKeys
	0x42	PNP_GetObjectProp
	0x43	PNP_SetObjectProp
	0x44	PNP_InstallDevInst
	0x45	PNP_ApplyPowerSettings
	0x46	PNP_DriverStoreAddDriverPackage
	0x47	PNP_DriverStoreDeleteDriverPackage
	0x48	PNP_RegisterServiceNotification
	0x49	PNP_SetActiveService
	0x4a	PNP_DeleteServiceDevices

In Windows NT 4.0, a similar interface exists with exactly the same interface identifier but in version 0.0 and with fewer operations (thanks to Derek Soeder for providing operations names).

```
Z:\>ifids -p ncacn_np -e \pipe\ntsvcs \\.
Interfaces: 7

[...]

  8d9f4e40-a03d-11ce-8f69-08003e30051b v0.0
```

Table 4.22. nt4_pnp operations

Interface	Operation number	Operation name
8d9f4e40-a03d-11ce-8f69-08003e30051b v0.0: nt4_pnp		
	0x00	PNP_Connect
	0x01	PNP_Disconnect
	0x02	PNP_GetVersion
	0x03	PNP_GetGlobalState
	0x04	PNP_InitDetection
	0x05	PNP_ReportLogOn

	0x06	PNP_ValidateDeviceInstance
	0x07	PNP_GetRootDeviceInstance
	0x08	PNP_GetRelatedDeviceInstance
	0x09	PNP_EnumerateSubKeys
	0x0a	PNP_GetDeviceList
	0x0b	PNP_GetDeviceListSize
	0x0c	PNP_GetDepth
	0x0d	PNP_GetDeviceRegProp
	0x0e	PNP_SetDeviceRegProp
	0x0f	PNP_GetClassInstance
	0x10	PNP_CreateKey
	0x11	PNP_DeleteRegistryKey
	0x12	PNP_GetClassCount
	0x13	PNP_GetClassName
	0x14	PNP_DeleteClassKey
	0x15	PNP_CreateDevInst
	0x16	PNP_DeviceInstanceAction
	0x17	PNP_GetDeviceStatus
	0x18	PNP_UninstallDevInst
	0x19	PNP_AddID
	0x1a	PNP_HwProfFlags
	0x1b	PNP_GetHwProfInfo
	0x1c	PNP_AddEmptyLogConf
	0x1d	PNP_FreeLogConf
	0x1e	PNP_GetFirstLogConf
	0x1f	PNP_GetNextLogConf
	0x20	PNP_AddResDes
	0x21	PNP_FreeResDes
	0x22	PNP_GetNextResDes
	0x23	PNP_GetResDesData
	0x24	PNP_GetResDesDataSize
	0x25	PNP_ModifyResDes

	0x26	PNP_DetectResourceConflict
	0x27	PNP_SetHwProf
	0x28	PNP_QueryArbitratorFreeData
	0x29	PNP_QueryArbitratorFreeSize
	0x2a	PNP_RunDetection

4.9.8. srvsvc interface

The **srvsvc** interface is used to manage the lanmanserver service.

IDL (Interface Definition Language) for the **srvsvc** interface is available in Samba 4 [59].

Table 4.23. srvsvc operations

Interface	Operation number	Operation name	Windows API
4b324fc8-1670-01d3-1278-5a47bf6ee188 v3.0: srvsvc			
	0x00	NetrCharDevEnum	NetCharDevEnum
	0x01	NetrCharDevGetInfo	NetCharDevGetInfo
	0x02	NetrCharDevControl	NetCharDevControl
	0x03	NetrCharDevQEnum	NetCharDevQEnum
	0x04	NetrCharDevQGetInfo	NetCharDevQGetInfo
	0x05	NetrCharDevQSetInfo	NetCharDevQSetInfo
	0x06	NetrCharDevQPurge	NetCharDevQPurge
	0x07	NetrCharDevQPurgeSelf	NetCharDevQPurgeSelf
	0x08	NetrConnectionEnum	NetConnectionEnum
	0x09	NetrFileEnum	NetFileEnum
	0x0a	NetrFileGetInfo	NetFileGetInfo
	0x0b	NetrFileClose	NetFileClose
	0x0c	NetrSessionEnum	NetSessionEnum
	0x0d	NetrSessionDel	NetSessionDel
	0x0e	NetrShareAdd	NetShareAdd
	0x0f	NetrShareEnum	NetShareEnum

	0x10	NetrShareGetInfo	NetShareGetInfo
	0x11	NetrShareSetInfo	NetShareSetInfo
	0x12	NetrShareDel	NetShareDel
	0x13	NetrShareDelSticky	NetShareDel
	0x14	NetrShareCheck	NetShareCheck
	0x15	NetrServerGetInfo	NetServerGetInfo
	0x16	NetrServerSetInfo	NetServerSetInfo
	0x17	NetrServerDiskEnum	NetServerDiskEnum
	0x18	NetrServerStatisticsGet	NetServerStatisticsGet
	0x19	NetrServerTransportAdd	NetServerTransportAdd
	0x1a	NetrServerTransportEnum	NetServerTransportEnum
	0x1b	NetrServerTransportDel	NetServerTransportDel
	0x1c	NetrRemoteTOD	NetRemoteTOD
	0x1d	NetrServerSetServiceBits	
	0x1e	NetprPathType	NetPathType
	0x1f	NetprPathCanonicalize	
	0x20	NetprPathCompare	
	0x21	NetprNameValidate	
	0x22	NetprNameCanonicalize	
	0x23	NetprNameCompare	
	0x24	NetrShareEnumSticky	NetShareEnum
	0x25	NetrShareDelStart	NetShareDel
	0x26	NetrShareDelCommit	NetShareDel
	0x27	NetrpGetFileSecurity	
	0x28	NetrpSetFileSecurity	
	0x29	NetrServerTransportAddEx	NetServerTransportAddEx
	0x2a	NetrServerSetServiceBitsEx	
	0x2b	NetrDfsGetVersion	
> Windows 2000	0x2c	NetrDfsCreateLocalPartition	
-	0x2d	NetrDfsDeleteLocalPartition	

-	0x2e	NetrDfsSetLocalVolumeState	
-	0x2f	NetrDfsSetServerInfo	
-	0x30	NetrDfsCreateExitPoint	
-	0x31	NetrDfsDeleteExitPoint	
-	0x32	NetrDfsModifyPrefix	
-	0x33	NetrDfsFixLocalVolume	
-	0x34	NetrDfsManagerReportSiteInfo	
> Windows XP and Windows Server 2003	0x35	NetrServerTransportDelEx	NetServerTransportDelEx
> Windows Vista	0x36	NetrServerAliasAdd	
	0x37	NetrServerAliasEnum	
	0x38	NetrServerAliasDel	
	0x39	NetrShareDelEx	

Obtaning general information on the server service:

- NetrServerGetInfo (0x15)
- NetrServerSetInfo (0x16)

Managing shares:

- NetrShareAdd (0x0e)
- NetrShareEnum (0x0f)
- NetrShareGetInfo (0x10)
- NetrShareSetInfo (0x11)
- NetrShareDel (0x12)
- NetrShareDelSticky (0x13)
- NetrShareCheck (0x14)

Managing established SMB sessions on a remote server:

- NetrSessionEnum (0x0c)
- NetrSessionDel (0x0d)

Managing opened files on a remote server:

- NetrFileEnum (0x09)
- NetrFileGetInfo (0x0a)

- NetrFileClose (0x0b)

Managing transports bindings of a remote SMB file server:

- NetrServerTransportAdd (0x19)
- NetrServerTransportEnum (0x1a)
- NetrServerTransportDel (0x1b)

Prev | Up | Next

4.9.7. pnp interface | Home | 4.9.9. svcctl interface

4.9.9. svcctl interface

The **svcctl** interface is used to manage Windows services via the SCM (Service Control Manager).

IDL (Interface Definition Language) for the **svcctl** interface is available in Samba 4 [60].

Table 4.24. svcctl operations

Interface	Operation number	Operation name	Windows API
367aeb81-9844-35f1-ad32-98f038001003 v2.0: svcctl			
	0x00	CloseServiceHandle	CloseServiceHandle
	0x01	ControlService	ControlService
	0x02	DeleteService	DeleteService
	0x03	LockServiceDatabase	LockServiceDatabase
	0x04	QueryServiceObjectSecurity	QueryServiceObjectSecurity
	0x05	SetServiceObjectSecurity	SetServiceObjectSecurity
	0x06	QueryServiceStatus	QueryServiceStatus
	0x07	SetServiceStatus	SetServiceStatus
	0x08	UnlockServiceDatabase	UnlockServiceDatabase
	0x09	NotifyBootConfigStatus	NotifyBootConfigStatus
	0x0a	ScSetServiceBitsW	SetServiceBits
	0x0b	ChangeServiceConfigW	ChangeServiceConfig
	0x0c	CreateServiceW	CreateService
	0x0d	EnumDependentServicesW	EnumDependentServices
	0x0e	EnumServicesStatusW	EnumServicesStatus
	0x0f	OpenSCManagerW	OpenSCManager

	0x10	OpenServiceW	OpenService
	0x11	QueryServiceConfigW	QueryServiceConfig
	0x12	QueryServiceLockStatusW	QueryServiceLockStatus
	0x13	StartServiceW	StartService
	0x14	GetServiceDisplayNameW	GetServiceDisplayName
	0x15	GetServiceKeyNameW	GetServiceKeyName
	0x16	ScSetServiceBitsA	SetServiceBits
	0x17	ChangeServiceConfigA	ChangeServiceConfig
	0x18	CreateServiceA	CreateService
	0x19	EnumDependentServicesA	EnumDependentServices
	0x1a	EnumServicesStatusA	EnumServicesStatus
	0x1b	OpenSCManagerA	OpenSCManager
	0x1c	OpenServiceA	OpenService
	0x1d	QueryServiceConfigA	QueryServiceConfig
	0x1e	QueryServiceLockStatusA	QueryServiceLockStatus
	0x1f	StartServiceA	StartService
	0x20	GetServiceDisplayNameA	GetServiceDisplayName
	0x21	GetServiceKeyNameA	GetServiceKeyName
	0x22	ScGetCurrentGroupStateW	
	0x23	EnumServiceGroupW	
	0x24	ChangeServiceConfig2A	ChangeServiceConfig2
	0x25	ChangeServiceConfig2W	ChangeServiceConfig2
	0x26	QueryServiceConfig2A	QueryServiceConfig2
> Windows 2000	0x27	QueryServiceConfig2W	QueryServiceConfig2
-	0x28	QueryServiceStatusEx	QueryServiceStatusEx
-	0x29	EnumServicesStatusExA	EnumServicesStatusEx
-	0x2a	EnumServicesStatusExW	EnumServicesStatusEx
> Windows XP and Windows Server 2003	0x2b	ScSendTSMessage	
> Windows Vista	0x2c	CreateServiceWOW64A	

	0x2d	CreateServiceWOW64W	
	0x2e	ScQueryServiceTagInfo	
	0x2f	NotifyServiceStatusChange	NotifyServiceStatusChange
	0x30	GetNotifyResult	NotifyServiceStatusChange
	0x31	CloseNotifyHandle	NotifyServiceStatusChange
	0x32	ControlServiceExA	ControlServiceExA
	0x33	ControlServiceExW	ControlServiceExW
	0x34	ScSendPnPMessage	
	0x35	ScValidatePnPService	
	0x36	ScOpenServiceStatusHandle	NotifyServiceStatusChange

The **svcctl** interface contains so-called Unicode operations (with names ending with W) and ASCII operations (with names ending with A).

Connecting to the SCM (Service Control Manager):

- OpenSCManagerW (0x0f)
- OpenSCManagerA (0x1b)

Locking and unlocking the SCM database:

- LockServiceDatabase (0x03)
- UnlockServiceDatabase (0x08)

Enumerating services:

- EnumServicesStatusW (0x0e)
- EnumServicesStatusExW (0x2a)
- EnumServicesStatusA (0x1a)
- EnumServicesStatusExA (0x29)

Obtaining a handle to a service:

- OpenServiceW (0x10)
- OpenServiceA (0x1c)

Starting a service:

- StartServiceW (0x13)
- StartServiceA (0x1f)

Obtaining the current status of a service:

- QueryServiceStatus (0x06)
- QueryServiceStatusEx (0x28)

Setting the current status of a service:

- SetServiceStatus (0x07)

Sending a control order to a service:

- ControlService (0x01)

Obtaining the detailed configuration of a service:

- QueryServiceConfigW (0x11)
- QueryServiceConfigA (0x1d)
- QueryServiceConfig2W (0x27)
- QueryServiceConfig2A (0x26)

Obtaining the current lock status of a service:

- QueryServiceLockStatusW (0x12)
- QueryServiceLockStatusA (0x1e)

Obtaining the display name of a service:

- GetServiceDisplayNameW (0x14)
- GetServiceDisplayNameA (0x20)

Obtaning the internal name of a service:

- GetServiceKeyNameW (0x15)
- GetServiceKeyNameA (0x21)

Modifying the configuration of a service:

- ChangeServiceConfigW (0x0b)

- ChangeServiceConfigA (0x17)
- ChangeServiceConfig2W (0x25)
- ChangeServiceConfig2A (0x24)

Examining and setting the security descriptor of a service:

- QueryServiceObjectSecurity (0x04)
- SetServiceObjectSecurity (0x05)

Adding a new service:

- CreateServiceW (0x0c)
- CreateServiceA (0x18)

Removing a service:

- DeleteService (0x02)

Enumerating service dependencies:

- EnumDependentServicesW (0x0d)
- EnumDependentServicesA (0x19)

Enumerating services that belong to a group:

- EnumServiceGroupW (0x23)

Closing a service or SCM handle:

- CloseServiceHandle (0x00)

4.9.10. winreg interface

The **winreg** interface is used to access to the registry, either locally or remotely. The interface also contains 3 operations related to systems shutdown.

IDL (Interface Definition Language) for the **winreg** interface is available in Samba 4 [61].

Table 4.25. winreg operations

Interface	Operation number	Operation name	Windows API
338cd001-2244-31f1-aaaa-900038001003 v1.0: winreg			
	0x00	OpenClassesRoot	RegConnectRegistry
	0x01	OpenCurrentUser	RegConnectRegistry
	0x02	OpenLocalMachine	RegConnectRegistry
	0x03	OpenPerformanceData	RegConnectRegistry
	0x04	OpenUsers	RegConnectRegistry
	0x05	BaseRegCloseKey	RegCloseKey
	0x06	BaseRegCreateKey	RegCreateKeyEx
	0x07	BaseRegDeleteKey	RegDeleteKeyEx
	0x08	BaseRegDeleteValue	RegDeleteValue
	0x09	BaseRegEnumKey	RegEnumKeyEx
	0x0a	BaseRegEnumValue	RegEnumValue
	0x0b	BaseRegFlushKey	RegFlushKey
	0x0c	BaseRegGetKeySecurity	RegGetKeySecurity
	0x0d	BaseRegLoadKey	RegLoadKey
	0x0e	BaseRegNotifyChangeKeyValue	RegNotifyChangeKeyValue

	0x0f	BaseRegOpenKey	RegOpenKeyEx
	0x10	BaseRegQueryInfoKey	RegQueryInfoKey
	0x11	BaseRegQueryValue	RegQueryValueEx
	0x12	BaseRegReplaceKey	RegReplaceKey
	0x13	BaseRegRestoreKey	RegRestoreKey
	0x14	BaseRegSaveKey	RegSaveKey
	0x15	BaseRegSetKeySecurity	RegSetKeySecurity
> Windows 2000	0x16	BaseRegSetValue	RegSetValueEx
-	0x17	BaseRegUnLoadKey	RegUnloadKey
-	0x18	BaseInitiateSystemShutdown	InitiateSystemShutdown
-	0x19	BaseAbortSystemShutdown	AbortSystemShutdown
-	0x1a	BaseRegGetVersion	
-	0x1b	OpenCurrentConfig	
-	0x1c	OpenDynData	
-	0x1d	BaseRegQueryMultipleValues	RegQueryMultipleValues
-	0x1e	BaseInitiateSystemShutdownEx	InitiateSystemShutdownEx
> Windows XP and Windows Server 2003	0x1f	BaseRegSaveKeyEx	RegSaveKeyEx
-	0x20	OpenPerformanceText	
-	0x21	OpenPerformanceNlsText	
> Windows Server 2003 SP1	0x22	BaseRegQueryMultipleValues2	
-	0x23	BaseRegDeleteKeyEx	RegDeleteKeyEx

4.9.11. wkssvc interface

The **wkssvc** interface is used to manage the lanmanworkstation service.

IDL (Interface Definition Language) for the **wkssvc** interface is available in Samba 4 [62].

Table 4.26. wkssvc operations

Interface	Operation number	Operation name	Windows API
6bffd098-a112-3610-9833-46c3f87e345a v1.0: wkssvc			
	0x00	NetrWkstaGetInfo	NetWkstaGetInfo
	0x01	NetrWkstaSetInfo	NetWkstaSetInfo
	0x02	NetrWkstaUserEnum	NetWkstaUserEnum
	0x03	NetrWkstaUserGetInfo	NetWkstaUserGetInfo
	0x04	NetrWkstaUserSetInfo	NetWkstaUserSetInfo
	0x05	NetrWkstaTransportEnum	NetWkstaTransportEnum
	0x06	NetrWkstaTransportAdd	NetWkstaTransportAdd
	0x07	NetrWkstaTransportDel	NetWkstaTransportDel
	0x08	NetrUseAdd	NetUseAdd
	0x09	NetrUseGetInfo	NetUseGetInfo
	0x0a	NetrUseDel	NetUseDel
	0x0b	NetrUseEnum	NetUseEnum
	0x0c	NetrMessageBufferSend	NetMessageBufferSend
	0x0d	NetrWorkstationStatisticsGet	NetWkstaStatisticsGet
	0x0e	NetrLogonDomainNameAdd	

> Windows 2000	0x0f	NetrLogonDomainNameDel	
-	0x10	NetrJoinDomain	NetJoinDomain
-	0x11	NetrUnjoinDomain	NetUnjoinDomain
-	0x12	NetrValidateName	NetValidateName
-	0x13	NetrRenameMachineInDomain	NetRenameMachineInDomain
-	0x14	NetrGetJoinInformation	NetGetJoinInformation
-	0x15	NetrGetJoinableOUs	NetGetJoinableOUs
-	0x16	NetrJoinDomain2	NetJoinDomain
-	0x17	NetrUnjoinDomain2	NetUnjoinDomain
-	0x18	NetrRenameMachineInDomain2	NetRenameMachineInDomain
-	0x19	NetrValidateName2	NetValidateName
-	0x1a	NetrGetJoinableOUs2	NetGetJoinableOUs
> Windows XP and Windows Server 2003	0x1b	NetrAddAlternateComputerName	NetAddAlternateComputerName
-	0x1c	NetrRemoveAlternateComputerName	NetRemoveAlternateComputerName
-	0x1d	NetrSetPrimaryComputerName	NetSetPrimaryComputerName
-	0x1e	NetrEnumerateComputerNames	NetEnumerateComputerNames
-	0x1f	NetrWorkstationResetDfsCache	

A vulnerability in the workstation service was discovered by Yuji Ukai [64] and fixed by Microsoft in November 2003 in the MS03-049 security bulletin [65]. It can be exploited anonymously because it is always possible to open the **wkssvc** named pipe in the context of a NULL session, as explained earlier.

Obtaining general information on the workstation service:

- NetrWkstaGetInfo (0x00)
- NetrWkstaSetInfo (0x01)

Managing SMB sessions (client-side):

- NetrUseEnum (0x0b)
- NetrUseGetInfo (0x09)

- NetrUseAdd (0x08)
- NetrUseDel (0x0a)

Managing transport mappings for the SMB redirector:

- NetrWkstaTransportEnum (0x05)
- NetrWkstaTransportAdd (0x06)
- NetrWkstaTransportDel (0x07)

Preparing a join operation:

- NetrGetJoinInformation (0x14)
- NetrGetJoinableOUs (0x15)
- NetrGetJoinableOUs2 (0x1a)

Joining or unjoining a remote machine to a domain:

- NetrJoinDomain (0x10)
- NetrJoinDomain2 (0x16)
- NetrUnjoinDomain (0x11)
- NetrUnjoinDomain2 (0x17)

Managing computer names:

- NetrAddAlternateComputerName (0x1b)
- NetrRemoveAlternateComputerName (0x1c)
- NetrSetPrimaryComputerName (0x1d)
- NetrEnumerateComputerNames (0x1e)

4.10. Windows services MSRPC interfaces

4.10.1. Active Directory domain controllers RPC services
4.10.2. Computer Browser service
4.10.3. DCOM Server Process Launcher
4.10.4. Distributed File System service
4.10.5. DNS server
4.10.6. Exchange RPC services
4.10.7. Exchange RPC services in Active Directory domains
4.10.8. File Replication service
4.10.9. IIS services
4.10.10. Inter-site Messaging service
4.10.11. Message Queuing and Distributed Transaction Coordinator services
4.10.12. Messenger service
4.10.13. NetDDE service
4.10.14. RPC locator service
4.10.15. Scheduler service
4.10.16. Spooler service
4.10.17. WINS service

4.10.1. Active Directory domain controllers RPC services

The following RPC interfaces are supported on a Windows 2000 domain controller to handle backup and restore of Active Directory:

```
Active Directory backup interface: ecec0d70-a603-11d0-96b1-00a0c91ece30 v1.0
Active Directory restore interface: 16e0cf3a-a604-11d0-96b1-00a0c91ece30 v1.0
```

Table 4.27. JetBack operations

Interface	Operation number	Operation name
ecec0d70-a603-11d0-96b1-00a0c91ece30 v1.0: JetBack		
	0x00	HrRBackupPrepare
	0x01	HrRBackupEnd
	0x02	HrRBackupGetAttachmentInformation
	0x03	HrRBackupOpenFile
	0x04	HrRBackupRead
	0x05	HrRBackupClose
	0x06	HrRBackupGetBackupLogs
	0x07	HrRBackupTruncateLogs
	0x08	HrRBackupPing

Table 4.28. JetRest operations

Interface	Operation number	Operation name
16e0cf3a-a604-11d0-96b1-00a0c91ece30 v1.0: JetRest		
	0x00	HrRIsNTDSOnline
	0x01	HrRRestorePrepare
	0x02	HrRRestoreRegister
	0x03	HrRRestoreRegisterComplete
	0x04	HrRRestoreGetDatabaseLocations
	0x05	HrRRestoreEnd
	0x06	HrRRestoreSetCurrentLogNumber

	0x07	HrRRestoreCheckLogsForBackup

By default, these RPC services are registered in the endpoint mapper database on a dynamic TCP port. However, it is possible to set a registry value to configure these services to listen on a fixed port [87]. Once this value is configured, the portmapper service will always return this fixed port when asked for one of these interfaces.

Windows Server 2003 and later versions support the dsrole interface, available on the following endpoint:

- dsrole LPC port

```
Y:\>ifids -p ncalrpc -e dsrole serveur
Interfaces: 18

[...]

  1cbcad78-df0b-4934-b558-87839ea501c9 v0.0

[...]
```

Table 4.29. dsrole operations

Interface	Operation number	Operation name
1cbcad78-df0b-4934-b558-87839ea501c9 v0.0: dsrole		
Windows Server 2003 and >	0x00	DsRolerDnsNameToFlatName
-	0x01	DsRolerDcAsDc
-	0x02	DsRolerDcAsReplica
-	0x03	DsRolerDemoteDc
-	0x04	DsRolerGetDcOperationProgress
-	0x05	DsRolerGetDcOperationResults
-	0x06	DsRolerCancel
-	0x07	DsRolerIfmHandleFree
-	0x08	DsRolerServerSaveStateForUpgrade
-	0x09	DsRolerUpgradeDownlevelServer
-	0x0a	DsRolerAbortDownlevelServerUpgrade
-	0x0b	DsRolerGetDatabaseFacts

There is another interface in the **ntdsa.dll** DLL, which contains only two operations:

Table 4.30. dsaop operations

Interface	Operation number	Operation name
7c44d7d4-31d5-424c-bd5e-2b3e1f323d22 v1.0: dsaop		
	0x00	DSAPrepareScript
	0x01	DSAExecuteScript

4.10.2. Computer Browser service

The Computer Browser service supports the **browser** interface:

Table 4.31. browser operations

Interface	Operation number	Operation name
6bffd098-a112-3610-9833-012892020162 v0.0: browser		
	0x00	BrowserrServerEnum
	0x01	BrowserrDebugCall
	0x02	BrowserrQueryOtherDomains
	0x03	BrowserrResetNetlogonState
	0x04	BrowserrDebugTrace
	0x05	BrowserrQueryStatistics
	0x06	BrowserrResetStatistics
	0x07	NetrBrowserStatisticsClear
	0x08	NetrBrowserStatisticsGet
	0x09	BrowserrSetNetlogonState
	0x0a	BrowserrQueryEmulatedDomains
	0x0b	BrowserrServerEnumEx

4.10.3. DCOM Server Process Launcher

Table 4.32. IActivationKernel operations

Interface	Operation number	Operation name
9b8699ae-0e44-47b1-8e7f-86a461d7ecdc v0.0: IActivationKernel		
	0x00	LaunchActivatorServer
	0x01	LaunchRunAsServer
	0x02	LaunchService
	0x03	CertifyServerIdentity
	0x04	PrivRunAsSetWinstaDesktop
	0x05	PrivRunAsRelease
	0x06	PrivRunAsInvalidateAndRelease
	0x07	PrivTranslateShareName
	0x08	FlushCache
	0x09	IsPortOpen
	0x0a	SignalPnpNotification

4.10.4. Distributed File System service

The Distributed File System service supports the **netdfs** interface.

IDL (Interface Definition Language) for the **netdfs** interface is available in Samba 4 [58].

Table 4.33. netdfs operations

Interface	Operation number	Operation name	Windows API
4fc742e0-4a10-11cf-8273-00aa004ae673 v3.0: netdfs			
	0x00	NetrDfsManagerGetVersion	
	0x01	NetrDfsAdd	NetDfsAdd
	0x02	NetrDfsRemove	NetDfsRemove
	0x03	NetrDfsSetInfo	NetDfsSetInfo
	0x04	NetrDfsGetInfo	NetDfsGetInfo
	0x05	NetrDfsEnum	NetDfsEnum
	0x06	NetrDfsRename	
	0x07	NetrDfsMove	NetDfsMove
	0x08	NetrDfsManagerGetConfigInfo	
	0x09	NetrDfsManagerSendSiteInfo	
	0x0a	NetrDfsAddFtRoot	NetDfsAddFtRoot
	0x0b	NetrDfsRemoveFtRoot	NetDfsRemoveFtRoot
	0x0c	NetrDfsAddStdRoot	NetDfsAddStdRoot
	0x0d	NetrDfsRemoveStdRoot	NetDfsRemoveStdRoot
	0x0e	NetrDfsManagerInitialize	NetDfsManagerInitialize
	0x0f	NetrDfsAddStdRootForced	NetDfsAddStdRootForced

	0x10	NetrDfsGetDcAddress	
	0x11	NetrDfsSetDcAddress	
	0x12	NetrDfsFlushFtTable	
	0x13	NetrDfsAdd2	
	0x14	NetrDfsRemove2	
	0x15	NetrDfsEnumEx	
	0x16	NetrDfsSetInfo2	

4.10.5. DNS server

Windows DNS server (**dns.exe** process) runs one RPC service, listening on the following endpoints:

- **DNSSERVERLPC** LPC port
- **dnsserver** named pipe
- a dynamic TCP port

```
Y:\>ifids -p ncalrpc -e DNSSERVERLPC serveur
Interfaces: 1
  50abc2a4-574d-40b3-9d66-ee4fd5fba076 v5.0

Y:\>ifids -p ncacn_np -e \pipe\dnsserver \\.
Interfaces: 1
  50abc2a4-574d-40b3-9d66-ee4fd5fba076 v5.0

Y:\>ifids -p ncacn_ip_tcp -e 3009 127.0.0.1
Interfaces: 1
  50abc2a4-574d-40b3-9d66-ee4fd5fba076 v5.0
```

It is possible to configure the RPC transports of the RPC service run by the DNS Server, using the **RpcProtocol** registry value, documented in Microsoft DNS Server Registry Parameters, Part 1 of 3:

```
Key: HKLM\SYSTEM\CCS\Services\DNS\Parameters\
Value: RpcProtocol (REG_DWORD)
Default value: 0xFFFFFFFF (all protocols enabled)
```

Table 4.34. DnsServer operations

Interface	Operation number	Operation name
50abc2a4-574d-40b3-9d66-ee4fd5fba076 v5.0: DnsServer		
	0x00	DnssrvOperation
	0x01	DnssrvQuery
	0x02	DnssrvComplexOperation

	0x03	DnssrvEnumRecords
	0x04	DnssrvUpdateRecord
> Windows XP and Windows Server 2003	0x05	DnssrvOperation2
-	0x06	DnssrvQuery2
-	0x07	DnssrvComplexOperation2
-	0x08	DnssrvEnumRecords2
-	0x09	DnssrvUpdateRecord2

Prev Up Next
4.10.4. Distributed File System service Home 4.10.6. Exchange RPC services

4.10.6. Exchange RPC services

The MAPI interface (also known as Exchange Server Store EMSMDB Interface) is identified as follows:

```
a4f1db00-ca47-1067-b31f-00dd010662da v0.81
```

Table 4.35. exchange_mapi operations

Interface	Operation number	Operation name
a4f1db00-ca47-1067-b31f-00dd010662da v0.81		
	0x00	EcDoConnect
	0x01	EcDoDisconnect
	0x02	EcDoRpc
	0x03	EcGetMoreRpc
	0x04	EcRRegisterPushNotification
	0x05	EcRUnregisterPushNotification
	0x06	EcDummyRpc
	0x07	EcRGetDCName
	0x08	EcRNetGetDCName
	0x09	EcDoRpcExt

[88] lists identifiers of Exchange RPC interfaces exposed when the Secure Mail Publishing feature of ISA Server 2000 is used.

The following interface identifiers are registered in the endpoint mapper database of an Exchange 2000 server:

```
Annotation=Exchange Server STORE ADMIN Interface
uuid=99e64010-b032-11d0-97a4-00c04fd6551d , version=3

annotation=Exchange Server STORE ADMIN Interface
```

```
uuid=89742ace-a9ed-11cf-9c0c-08002be7ae86 , version=2

annotation=Exchange Server STORE ADMIN Interface
uuid=a4f1db00-ca47-1067-b31e-00dd010662da , version=1

annotation=Exchange Server STORE EMSMDB Interface
uuid=a4f1db00-ca47-1067-b31f-00dd010662da , version=0

annotation=MS Exchange MTA 'Mta' Interface
uuid=9e8ee830-4459-11ce-979b-00aa005ffebe , version=2

annotation=MS Exchange Directory NSPI Proxy
uuid=f5cc5a18-4264-101a-8c59-08002b2f8426 , version=56

annotation=MS Exchange MTA 'QAdmin' Interface
uuid=38a94e72-a9bc-11d2-8faf-00c04fa378ff , version=1

annotation=Microsoft Information Store
uuid=0e4a0156-dd5d-11d2-8c2f-00c04fb6bcde , version=1

annotation=Microsoft Information Store
uuid=1453c42c-0fa6-11d2-a910-00c04f990f3b , version=1

annotation=Microsoft Information Store
uuid=10f24e8e-0fa6-11d2-a910-00c04f990f3b , version=1

annotation=MS Exchange Directory RFR Interface
uuid=1544f5e0-613c-11d1-93df-00c04fd7bd09 , version=1

annotation=MS Exchange System Attendant Cluster Interface
uuid=f930c514-1215-11d3-99a5-00a0c9b61b04 , version=1

annotation=MS Exchange System Attendant Private Interface
uuid=83d72bf0-0d89-11ce-b13f-00aa003bac6c , version=6

annotation=MS Exchange System Attendant Public Interface
uuid=469d6ec0-0d87-11ce-b13f-00aa003bac6c , version=16
```

Table 4.36. exchange_rfr operations

Interface	Operation number	Operation name

1544f5e0-613c-11d1-93df-00c04fd7bd09 v1.0		
	0x00	RfrGetNewDSA
	0x01	RfrGetFQDNFromLegacyDN

4.10.7. Exchange RPC services in Active Directory domains

Active Directory domain controllers that have the Global Catalog server roles register the following RPC services, which are used by MAPI clients to access the Directory Service that was previously integrated in Exchange before Exchange 2000:

```
Active Directory Extended Directory Service (XDS): f5cc5a7c-4264-101a-8c59-
08002b2f8426 v21.0
Active Directory Name Service Provider (NSP) interface: f5cc5a18-4264-101a-8c59-
08002b2f8426 v56.0
```

Table 4.37. rxds operations

Interface	Operation number	Operation name
f5cc5a7c-4264-101a-8c59-08002b2f8426 v21.0: rxds		
	0x00	ds_abandon
	0x01	ds_add_entry
	0x02	ds_bind
	0x03	ds_compare
	0x04	ds_list
	0x05	ds_modify_entry
	0x06	ds_modify_rdn
	0x07	ds_read
	0x08	ds_receive_result
	0x09	ds_remove_entry
	0x0a	ds_search
	0x0b	ds_unbind
	0x0c	ds_wait
	0x0d	dra_replica_add
	0x0e	dra_replica_delete
	0x0f	dra_replica_synchronize
	0x10	dra_reference_update
	0x11	dra_authorize_replica
	0x12	dra_unauthorize_replica
	0x13	dra_adopt

	0x14	dra_set_status
	0x15	dra_modify_entry
	0x16	dra_delete_subref

Table 4.38. nspi operations

Interface	Operation number	Operation name
f5cc5a18-4264-101a-8c59-08002b2f8426 v56.0: nspi		
	0x00	NspiBind
	0x01	NspiUnbind
	0x02	NspiUpdateStat
	0x03	NspiQueryRows
	0x04	NspiSeekEntries
	0x05	NspiGetMatches
	0x06	NspiResortRestriction
	0x07	NspiDNToEph
	0x08	NspiGetPropList
	0x09	NspiGetProps
	0x0a	NspiCompareDNTs
	0x0b	NspiModProps
	0x0c	NspiGetHierarchyInfo
	0x0d	NspiGetTemplateInfo
	0x0e	NspiModLinkAtt
	0x0f	NspiDeleteEntries
	0x10	NspiQueryColumns
	0x11	NspiGetNamesFromIDs
	0x12	NspiGetIDsFromNames
	0x13	NspiResolveNames
	0x14	NspiResolveNamesW

NSPI operations offered by an Global Catalog Active Directory domain controller are either called directly (Outlook 2000 and later MAPI clients) or through a proxy run by the Exchange server, as described in [89].

An Exchange server integrated in an Active Directory domain registers the NSPI interface, to proxy NSPI requests to Global Catalog Active Directory domain controllers:

```
annotation=MS Exchange Directory NSPI Proxy
uuid=f5cc5a18-4264-101a-8c59-08002b2f8426 , version=56
ncacn_ip_tcp:172.16.1.238[1112]
```

```
annotation=MS Exchange Directory NSPI Proxy
uuid=f5cc5a18-4264-101a-8c59-08002b2f8426 , version=56
ncacn_http:172.16.1.238[1113]
```

The rxds interface is also registered on an Exchange 2000 server but is not registered in the endpoint mapper:

```
f5cc5a7c-4264-101a-8c59-08002b2f8426 v21.0
```

Prev	Up	Next
4.10.6. Exchange RPC services	Home	4.10.8. File Replication service

4.10.8. File Replication service

The File Replication Service (**ntfrs.exe** process) runs 3 RPC services on one TCP port:

```
f5cc59b4-4264-101a-8c59-08002b2f8426 v1.1
d049b186-814f-11d1-9a3c-00c04fc9b232 v1.1
a00c021c-2be2-11d2-b678-0000f87a8f8e v1.0
```

Table 4.39. FrsRpc operations

Interface	Operation number	Operation name
f5cc59b4-4264-101a-8c59-08002b2f8426 v1.1: FrsRpc		
	0x00	FrsRpcSendCommPkt
	0x01	FrsRpcVerifyPromotionParent
	0x02	FrsRpcStartPromotionParent
	0x03	FrsNOP
	0x04	FrsBackupComplete
	0x05	FrsBackupComplete
	0x06	FrsBackupComplete
	0x07	FrsBackupComplete
	0x08	FrsBackupComplete
	0x09	FrsBackupComplete
	0x0a	FrsRpcVerifyPromotionParentEx

Table 4.40. NtFrsApi operations

Interface	Operation number	Operation name
d049b186-814f-11d1-9a3c-00c04fc9b232 v1.1: NtFrsApi		
	0x00	VerifyPromotion

	0x01	PromotionStatus
	0x02	StartDemotion
	0x03	CommitDemotion
	0x04	Set_DsPollingIntervalW
	0x05	Get_DsPollingIntervalW
	0x06	VerifyPromotionW
	0x07	InfoW
	0x08	IsPathReplicated
	0x09	WriterCommand
	0x0a	ForceReplication

Table 4.41. PerfFrs operations

Interface	Operation number	Operation name
a00c021c-2be2-11d2-b678-0000f87a8f8e v1.0: PerfFrs		
	0x00	GetIndicesOfInstancesFromServer
	0x01	GetCounterDataOfInstancesFromServer

Prev | Up | Next

4.10.7. Exchange RPC services in Active Directory domains | Home | 4.10.9. IIS services

4.10.9. IIS services

In Windows 2000, IIS (Internet Information Server) 5 services (HTTP, SMTP, FTP, NNTP) run in a single process, **inetinfo.exe**.

The **inetinfo.exe** (IIS 5) process runs RPC services on the following endpoints:

- **INETINFO_LPC** LPC port
- **INETINFO** named pipe
- one dynamic TCP port and one dynamic UDP port

The following RPC service is registered by the IISAdmin service (**infocomm.dll**):

```
82ad4280-036b-11cf-972c-00aa006887b0 v2.0: inetinfo
```

Table 4.42. inetinfo operations

Interface	Operation number	Operation name
82ad4280-036b-11cf-972c-00aa006887b0 v2.0: inetinfo		
	0x00	_R_InetInfoGetVersion
	0x01	_R_InetInfoGetAdminInformation
	0x02	_R_InetInfoGetSites
	0x03	_R_InetInfoSetAdminInformation
	0x04	_R_InetInfoGetGlobalAdminInformation
	0x05	_R_InetInfoSetGlobalAdminInformation
	0x06	_R_InetInfoQueryStatistics
	0x07	_R_InetInfoClearStatistics
	0x08	_R_InetInfoFlushMemoryCache
	0x09	_R_InetInfoGetServerCapabilities
	0x0a	_R_W3QueryStatistics2
	0x0b	_R_W3ClearStatistics2

	0x0c	_R_FtpQueryStatistics2
	0x0d	_R_FtpClearStatistics2
	0x10	_R_IISEnumerateUsers
	0x11	_R_IISDisconnectUser
	0x12	_R_InitW3CounterStructure
	0x13	_R_CollectW3PerfData

The SMTP service (**smtpsvc.dll**) runs the following RPC service:

```
8cfb5d70-31a4-11cf-a7d8-00805f48a135 v3.0
```

Table 4.43. iis_smtp operations

Interface	Operation number	Operation name
8cfb5d70-31a4-11cf-a7d8-00805f48a135 v3.0		
	0x00	SmtprGetAdminInformation
	0x01	SmtprSetAdminInformation
	0x02	SmtprQueryStatistics
	0x03	SmtprClearStatistics
	0x04	SmtprGetConnectedUserList
	0x05	SmtprDisconnectUser
	0x06	SmtprCreateUser
	0x07	SmtprDeleteUser
	0x08	SmtprGetUserProps
	0x09	SmtprSetUserProps
	0x0a	SmtprCreateDistList
	0x0b	SmtprDeleteDistList
	0x0c	SmtprCreateDistListMember
	0x0d	SmtprDeleteDistListMember
	0x0e	SmtprGetNameList
	0x0f	SmtprGetNameListFromList
	0x10	SmtprGetVRootSize
	0x11	SmtprBackupRoutingTable

The NNTP service (**ntpsvc.dll**) runs the following RPC service:

`4f82f460-0e21-11cf-909e-00805f48a135 v4.0`

Table 4.44. iis_nntp operations

Interface	Operation number	Operation name
4f82f460-0e21-11cf-909e-00805f48a135 v4.0		
	0x00	NntprQueryStatistic
	0x01	NntprClearStatistics
	0x02	NntprEnumerateFeeds
	0x03	NntprGetFeedInformation
	0x04	NntprSetFeedInformation
	0x05	NntprAddFeed
	0x06	NntprDeleteFeed
	0x07	NntprEnableFeed
	0x08	NntprEnumerateSessions
	0x09	NntprTerminateSession
	0x0a	NntprEnumerateExpires
	0x0b	NntprAddExpire
	0x0c	NntprDeleteExpire
	0x0d	NntprGetExpireInformation
	0x0e	NntprSetExpireInformation
	0x0f	NntprGetNewsgroup
	0x10	NntprSetNewsgroup
	0x11	NntprCreateNewsgroup
	0x12	NntprDeleteNewsgroup
	0x13	NntprFindNewsgroup
	0x14	NntprGetAdminInformation
	0x15	NntprSetAdminInformation
	0x16	NntprStartRebuild
	0x17	NntprGetBuildStatus

	0x18	NntprCancelMessageID
	0x19	NntprGetVRootWin32Error

The IMAP4 service (**imap4svc.dll**), installed by Exchange, runs the following RPC service:

```
2465e9e0-a873-11d0-930b-00a0c90ab17c v3.0
```

Table 4.45. iis_imap operations

Interface	Operation number	Operation name
2465e9e0-a873-11d0-930b-00a0c90ab17c v3.0		
	0x00	ImaprQueryStatistics
	0x01	ImaprClearStatistics
	0x02	ImaprGetConnectedUserList
	0x03	ImaprDisconnectUser

The POP3 service (**pop3svc.dll**), installed by Exchange, runs the following RPC service:

```
1be617c0-31a5-11cf-a7d8-00805f48a135 v3.0
```

Table 4.46. iis_pop operations

Interface	Operation number	Operation name
1be617c0-31a5-11cf-a7d8-00805f48a135 v3.0		
	0x00	Pop3rQueryStatistics
	0x01	Pop3rClearStatistics
	0x02	Pop3rGetConnectedUserList
	0x03	Pop3rDisconnectUser

The following interface identifiers correspond to the GUID of the COM components activated to handle IIS management :

```
70b51430-b6ca-11d0-b9b9-00a0c922e750 v0.0: IMSAdminBaseW
a9e69612-b80d-11d0-b9b9-00a0c922e750 v0.0: IADMCOMSINK
```

4.10.10. Inter-site Messaging service

The Inter-site Messaging service (**ismserv.exe** process) runs one RPC service, available on the following endpoints:

- **ISMSERV_LPC** LPC port

```
Y:\>ifids -p ncalrpc -e ISMSERV_LPC serveur
Interfaces: 1
  68dcd486-669e-11d1-ab0c-00c04fc2dcd2 v2.0
```

Table 4.47. ismapi operations

Interface	Operation number	Operation name
68dcd486-669e-11d1-ab0c-00c04fc2dcd2 v1.0: ismapi		
	0x00	ISMSend
	0x01	ISMReceive
	0x02	ISMGetConnectivity
	0x03	ISMGetTransportServers
	0x04	ISMGetConnectionSchedule
	0x05	ISMQuerySitesByCost

The following RPC service runs in the **ismip.dll** DLL, loaded in the **ismserv.exe** process context:

```
Active Directory ISM IP Transport: 130ceefb-e466-11d1-b78b-00c04fa32883 v2.1
```

This interface contains only one operation:

Table 4.48. ismserv_ip operations

Interface	Operation number	Operation name
130ceefb-e466-11d1-b78b-00c04fa32883 v2.1		
	0x00	ISMXXX

4.10.11. Message Queuing and Distributed Transaction Coordinator services

The Message Queuing service (msmq) runs RPC services, listening on the **ncacn_ip_tcp** transport. By default, the msmq services opens 4 TCP ports [81], including one or several of 2101/tcp, 2103/tcp, 2105/tcp and 2107/tcp.

The **mqqm.dll** (Windows NT MQ Queue Manager) DLL, loaded in the **mqsvc.exe** process, contains the following RPC services:

```
fdb3a030-065f-11d1-bb9b-00a024ea5525 v1.0
76d12b80-3467-11d3-91ff-0090272f9ea3 v1.0
1088a980-eae5-11d0-8d9b-00a02453c337 v1.0
5b5b3580-b0e0-11d1-b92d-0060081e87f0 v1.0
41208ee0-e970-11d1-9b9e-00e02c064c39 v1.0
```

Table 4.49. qmcomm operations

Interface	Operation number	Operation name
fdb3a030-065f-11d1-bb9b-00a024ea5525 v1.0: qmcomm		
	0x00	QMOpenQueue
	0x01	QMGetRemoteQueueName
	0x02	QMOpenRemoteQueue
	0x03	QMCloseRemoteQueueContext
	0x04	QMCreateRemoteCursor
	0x05	QMSendMessageInternal
	0x06	QMCreateObjectInternal
	0x07	QMSetObjectSecurityInternal
	0x08	QMGetObjectSecurityInternal
	0x09	QMDeleteObject
	0x0a	QMGetObjectProperties
	0x0b	QMSetObjectProperties

	0x0c	QMObjectPathToObjectFormat
	0x0d	QMAttachProcess
	0x0e	QMGetTmWhereabouts
	0x0f	QMEnlistTransation
	0x10	QMEnlistInternalTransaction
	0x11	QMCommitTransaction
	0x12	QMAbortTransaction
	0x13	QMOpenQueueInternal
	0x14	ACCloseHandle
	0x15	ACCreateCursor
	0x16	ACCloseCursor
	0x17	ACSetCursorProperties
	0x18	ACSendMessage
	0x19	ACReceiveMessage
	0x1a	ACHandleToFormatName
	0x1b	ACPurgeQueue
	0x1c	QMQueryQMRegistryInternal
	0x1d	QMListInternalQueues
	0x1e	QMCorrectOutSequence
	0x1f	QMGetRemoteQMServerPort
	0x20	QMGetMsmqServiceName
	0x21	QMCreateDSObjectInternal

A vulnerability in the QMDeleteObject operation was discovered by Kostya Kortchinsky and fixed by the MS05-017 security bulletin [82] in April 2005.

Table 4.50. qmcomm2 operations

Interface	Operation number	Operation name
76d12b80-3467-11d3-91ff-0090272f9ea3 v1.0: qmcomm2		
	0x00	QMSendMessageInternalEx
	0x01	ACSendMessageEx

	0x02	ACReceiveMessageEx
	0x03	ACCreateCursorEx

Table 4.51. qm2qm operations

Interface	Operation number	Operation name
1088a980-eae5-11d0-8d9b-00a02453c337 v1.0: qm2qm		
	0x00	RemoteQMStartReceive
	0x01	RemoteQMEndReceive
	0x02	RemoteQMOpenQueue
	0x03	RemoteQMCloseQueue
	0x04	RemoteQMCloseCursor
	0x05	RemoteQMCancelReceive
	0x06	RemoteQMPurgeQueue
	0x07	RemoteQMGetQMQMServerPort
	0x08	RemoteQmGetVersion
	0x09	RemoteQMStartReceive2
	0x0a	RemoteQMStartReceiveByLookupId

Table 4.52. qmrepl operations

Interface	Operation number	Operation name
5b5b3580-b0e0-11d1-b92d-0060081e87f0 v1.0: qmrepl		
	0x00	QMSendReplMsg

Table 4.53. qmmgmt operations

Interface	Operation number	Operation name
41208ee0-e970-11d1-9b9e-00e02c064c39 v1.0: qmmgmt		
	0x00	QMMgmtGetInfo
	0x01	QMMgmtAction

The **msdtcprx.dll** (MS DTC OLE Transactions interface proxy) DLL, also loaded in the **mqsvc.exe** process, also contains one RPC service:

`906b0ce0-c70b-1067-b317-00dd010662da v1.0`

Table 4.54. IXnRemote operations

Interface	Operation number	Operation name
906b0ce0-c70b-1067-b317-00dd010662da v1.0: IXnRemote		
	0x00	Poke
	0x01	BuildContext
	0x02	NegotiateResources
	0x03	SendReceive
	0x04	TearDownContext
	0x05	BeginTearDown
	0x06	PokeW
	0x07	BuildContextW

This RPC service also runs in the Distributed Transaction Coordinator service process (**msdtc.exe**), which opens a dynamic port, as well as TCP port 3372 (at least on Windows 2000)

4.10.12. Messenger service

The messenger service runs two RPC services, available on two endpoints:

- **msgsvc** named pipe
- a dynamic UDP port

The MS03-043 Microsoft security bulletin removed support for the **msgsvcsend** interface and for the UDP endpoint, leaving only the **msgsvc** named pipe.

Because the Messenger service is running in a shared process, removing the UDP endpoint was an important security improvement because before, the **ncadg_ip_udp** transport could be used with this endpoint to reach anonymously other RPC services running in the same process.

Windows XP SP2 and Windows Server 2003 SP1 do not support the **msgsvcsend** interface and thus do not have the UDP endpoint. In addition, the Messenger service is disabled by default on Windows Server 2003 (all versions) and Windows XP SP2.

```
Y:\>ifids -p ncacn_np -e \pipe\msgsvc \\.
Interfaces: 42

[...]

  17fdd703-1827-4e34-79d4-24a55c53bb37 v1.0
  5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc v1.0

Y:\>ifids -p ncadg_ip_udp -e 4870 127.0.0.1
Interfaces: 42

[...]

  17fdd703-1827-4e34-79d4-24a55c53bb37 v1.0
  5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc v1.0
```

The UDP transport is frequently used with the **msgsvcsend** interface to massively send popup windows containing advertisement messages [77].

The two RPC services run by the messenger service have the following interfaces identifiers:

```
17fdd703-1827-4e34-79d4-24a55c53bb37 v1.0: msgsvc
5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc v1.0: msgsvcsend
```

The **msgsvc** interface supports 4 operations that manipulate NetBIOS names on a local or remote system:

Table 4.55. msgsvc operations

Interface	Operation number	Operation name
17fdd703-1827-4e34-79d4-24a55c53bb37 v1.0: msgsvc		
	0x00	NetrMessageNameAdd
	0x01	NetrMessageNameEnum
	0x02	NetrMessageNameGetInfo
	0x03	NetrMessageNameDel

The **msgsvcsend** interface supports one operation, to send a message to a registered NetBIOS name using MSRPC:

Table 4.56. msgsvcsend operation

Interface	Operation number	Operation name
5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc v1.0: msgsvcsend		
	0x00	NetrSendMessage

The **msgsvcsend** interface is frequently used to send advertisement messages, using the **NetrSendMessage** operation.

The MS03-043 [78] Microsoft security bulletin includes a patch that completely removes support for the **msgsvcsend** interface of the Messenger service (both server-side function in **msgsvc.dll** and client-side function in **wkssvc.dll** are removed in patched versions of these two DLL).

4.10.13. NetDDE service

Table 4.57. nddeapi operations

Interface	Operation number	Operation name
2f5f3220-c126-1076-b549-074d078619da v1.2: nddeapi		
	0x00	NDdeShareAddW
	0x01	NDdeShareDelA
	0x02	NDdeShareDelW
	0x03	NDdeGetShareSecurityA
	0x04	NDdeGetShareSecurityW
	0x05	NDdeSetShareSecurityA
	0x06	NDdeSetShareSecurityW
	0x07	NDdeShareEnumA
	0x08	NDdeShareEnumW
	0x09	NDdeShareGetInfoW
	0x0a	NDdeShareSetInfoW
	0x0b	NDdeSetTrustedShareA
	0x0c	NDdeSetTrustedShareW
	0x0d	NDdeGetTrustedShareA
	0x0e	NDdeGetTrustedShareW
	0x0f	NDdeTrustedShareEnumA
	0x10	NDdeTrustedShareEnumW
	0x12	NDdeSpecialCommand

4.10.14. RPC locator service

The RPC locator service runs one RPC service, available on the following endpoint:

- **locator** named pipe

```
Y:\>ifids -p ncacn_np -e \pipe\locator \\.
Interfaces: 3
  d6d70ef0-0e3b-11cb-acc3-08002b1d29c3 v1.0
  d3fbb514-0e3b-11cb-8fad-08002b1d29c3 v1.0
  d6d70ef0-0e3b-11cb-acc3-08002b1d29c4 v1.0
```

Table 4.58. NsiS operations

Interface	Operation number	Operation name
d6d70ef0-0e3b-11cb-acc3-08002b1d29c3 v1.0: NsiS		
	0x00	nsi_binding_export
	0x01	nsi_binding_unexport

Table 4.59. NsiC operations

Interface	Operation number	Operation name
d3fbb514-0e3b-11cb-8fad-08002b1d29c3 v1.0: NsiC		
	0x00	nsi_binding_lookup_begin
	0x01	nsi_binding_lookup_done
	0x02	nsi_binding_lookup_next
	0x03	nsi_mgmt_handle_set_exp_age

Table 4.60. NsiM operations

Interface	Operation number	Operation name

d6d70ef0-0e3b-11cb-acc3-08002b1d29c4 v1.0: NsiM		
	0x00	nsi_group_delete
	0x01	nsi_group_mbr_add
	0x02	nsi_group_mbr_remove
	0x03	nsi_group_mbr_inq_begin
	0x04	nsi_group_mbr_inq_next
	0x05	nsi_group_mbr_inq_done
	0x06	nsi_profile_delete
	0x07	nsi_profile_elt_add
	0x08	nsi_profile_elt_remove
	0x09	nsi_profile_elt_inq_begin
	0x0a	nsi_profile_elt_inq_next
	0x0b	nsi_profile_elt_inq_done
	0x0c	nsi_entry_object_inq_begin
	0x0d	nsi_entry_object_inq_next
	0x0e	nsi_entry_object_inq_done
	0x0f	nsi_entry_expand_name
	0x10	nsi_mgmt_binding_unexport
	0x11	nsi_mgmt_entry_delete
	0x12	nsi_mgmt_entry_create
	0x13	nsi_mgmt_entry_inq_if_ids
	0x14	nsi_mgmt_inq_exp_age
	0x15	nsi_mgmt_inq_set_age

A vulnerability in the locator service was published by David Litchfield in January 2003 [75]. It was fixed by the MS03-001 Microsoft security patch [76].

As the **locator** named pipe is one of the named pipe that can be accessed in the context of a NULL session, this vulnerability can be exploited remotely without any authentication.

4.10.15. Scheduler service

The scheduler service runs RPC services allowing remote configuration of scheduled tasks (AT jobs). These RPC services are available on two endpoints:

- **atsvc** named pipe
- A dynamic TCP port (removed in Windows XP SP2 and Windows Server 2003 SP1)

Before Windows XP the Scheduler service was implemented in a single process, **mstask.exe**. Starting with Windows XP, the Scheduler service runs in a **svchost.exe** instance process (**schedsvc.dll**) and runs an additional RPC service (the third one in the list below).

The interfaces identifiers of these RPC services are:

```
X:\>ifids -p ncacn_np -e \pipe\atsvc \\.
Interfaces: 51

[...]

  1ff70682-0a51-30e8-076d-740be8cee98b v1.0
  378e52b0-c0a9-11cf-822d-00aa0051e40f v1.0
  0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 v1.0

X:\>ifids -p ncacn_ip_tcp -e 3136 127.0.0.1
Interfaces: 51

[...]

  1ff70682-0a51-30e8-076d-740be8cee98b v1.0
  378e52b0-c0a9-11cf-822d-00aa0051e40f v1.0
  0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 v1.0

X:\>
```

IDL (Interface Definition Language) for the **atsvc** interface is available in Samba 4 [79].

Table 4.61. atsvc operations

Interface	Operation number	Operation name
1ff70682-0a51-30e8-076d-740be8cee98b v1.0: atsvc		
	0x00	NetrJobAdd
	0x01	NetrJobDel
	0x02	NetrJobEnum
	0x03	NetrJobGetInfo

Enumerating AT jobs:

- NetrJobEnum (0x02)

Adding or removing an AT job:

- NetrJobAdd (0x00)
- NetrJobDel (0x01)

Obtaining details about an AT job:

- NetrJobGetInfo (0x03)

Submission of AT jobs is by default restricted to only members of the Administrators group. On domain controllers, it is possible to also allow members of the Server Operators group to submit AT jobs, by setting the SubmitControl registry value to 1 (not recommended).

Table 4.62. sasec operations

Interface	Operation number	Operation name
378e52b0-c0a9-11cf-822d-00aa0051e40f v1.0: sasec		
	0x00	SASetAccountInformation
	0x01	SASetNSAccountInformation
	0x02	SAGetNSAccountInformation
	0x03	SAGetAccountInformation

The **idletask** interface was added in Windows XP:

Table 4.63. idletask operations

Interface	Operation number	Operation name
0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 v1.0: idletask		
	0x00	ItSrvRegisterIdleTask
	0x01	ItSrvUnregisterIdleTask
	0x02	ItSrvProcessIdleTasks
	0x03	ItSrvSetDetectionParameters

In Windows Vista, a new interface, **ITaskSchedulerService**, was added

```
Y:\>ifids -p ncacn_np -e \pipe\atsvc \\.
Interfaces: 56

[...]

  0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 v1.0
  1ff70682-0a51-30e8-076d-740be8cee98b v1.0
  378e52b0-c0a9-11cf-822d-00aa0051e40f v1.0
  86d35949-83c9-4044-b424-db363231fd0c v1.0

[...]
```

Table 4.64. ITaskSchedulerService operations

Interface	Operation number	Operation name
86d35949-83c9-4044-b424-db363231fd0c v1.0: ITaskSchedulerService		
	0x00	SchRpcHighestVersion
	0x01	SchRpcRegisterTask
	0x02	SchRpcRetrieveTask
	0x03	SchRpcCreateFolder
	0x04	SchRpcSetSecurity
	0x05	SchRpcGetSecurity

	0x06	SchRpcEnumFolder
	0x07	SchRpcEnumTasks
	0x08	SchRpcEnumInstances
	0x09	SchRpcGetInstanceInfo
	0x0a	SchRpcStopInstance
	0x0b	SchRpcStop
	0x0c	SchRpcRun
	0x0d	SchRpcDelete
	0x0e	SchRpcRename
	0x0f	SchRpcScheduledRuntimes
	0x10	SchRpcGetLastRunInfo
	0x11	SchRpcGetTaskInfo

4.10.16. Spooler service

The Spooler service runs one RPC service, **spoolss**:

```
Z:\>ifids -p ncacn_np -e \pipe\spoolss \\.
Interfaces: 1
  12345678-1234-abcd-ef00-0123456789ab v1.0

Z:\>ifids -p ncalrpc-e spoolss serveur
Interfaces: 1
  12345678-1234-abcd-ef00-0123456789ab v1.0
```

Starting with Windows Server 2003, the Spooler service does not create the **spoolss** named pipe endpoint by default if no shared printer is configured. Instead, the **spoolss** LPC port is used as local endpoint to communicate with the Spooler service.

It is possible to set the **RegisterSpoolerRemoteRpcEndpoint** registry value to 1 to force the creation of the **spoolss** named pipe endpoint, even if no shared printer is configured:

```
GPO: Allow Print Spooler to accept client connections
Key: HKLM\Software\Policies\Microsoft\Windows NT\Printers
Value: RegisterSpoolerRemoteRpcEndPoint (REG_DWORD)
Default value: 0
```

IDL (Interface Definition Language) for the **spoolss** interface is available in Samba 4 [66].

Table 4.65. winspool operations

Interface	Operation number	Operation name	Windows API
12345678-1234-abcd-ef00-0123456789ab v1.0: winspool (spoolss)			
	0x00	RpcEnumPrinters	EnumPrinters
	0x01	RpcOpenPrinter	OpenPrinter
	0x02	RpcSetJob	SetJob

	0x03	RpcGetJob	GetJob
	0x04	RpcEnumJobs	EnumJobs
	0x05	RpcAddPrinter	AddPrinter
	0x06	RpcDeletePrinter	DeletePrinter
	0x07	RpcSetPrinter	SetPrinter
	0x08	RpcGetPrinter	GetPrinter
	0x09	RpcAddPrinterDriver	AddPrinterDriver
	0x0a	RpcEnumPrinterDrivers	EnumPrinterDrivers
	0x0b	RpcGetPrinterDriver	GetPrinterDriver
	0x0c	RpcGetPrinterDriverDirectory	GetPrinterDriverDirectory
	0x0d	RpcDeletePrinterDriver	DeletePrinterDriver
	0x0e	RpcAddPrintProcessor	AddPrintProcessor
	0x0f	RpcEnumPrintProcessors	EnumPrintProcessors
	0x10	RpcGetPrintProcessorDirectory	GetPrintProcessorDirectory
	0x11	RpcStartDocPrinter	StartDocPrinter
	0x12	RpcStartPagePrinter	StartPagePrinter
	0x13	RpcWritePrinter	WritePrinter
	0x14	RpcEndPagePrinter	EndPagePrinter
	0x15	RpcAbortPrinter	AbortPrinter
	0x16	RpcReadPrinter	ReadPrinter
	0x17	RpcEndDocPrinter	EndDocPrinter
	0x18	RpcAddJob	AddJob
	0x19	RpcScheduleJob	ScheduleJob
	0x1a	RpcGetPrinterData	GetPrinterData
	0x1b	RpcSetPrinterData	SetPrinterData
	0x1c	RpcWaitForPrinterChange	
	0x1d	RpcClosePrinter	ClosePrinter
	0x1e	RpcAddForm	AddForm
	0x1f	RpcDeleteForm	DeleteForm
	0x20	RpcGetForm	GetForm
	0x21	RpcSetForm	SetForm
	0x22	RpcEnumForms	EnumForms
	0x23	RpcEnumPorts	EnumPorts

	0x24	RpcEnumMonitors	EnumMonitors
	0x25	RpcAddPort	AddPort
	0x26	RpcConfigurePort	ConfigurePort
	0x27	RpcDeletePort	DeletePort
	0x28	RpcCreatePrinterIC	
	0x29	RpcPlayGdiScriptOnPrinterIC	
	0x2a	RpcDeletePrinterIC	
	0x2b	RpcAddPrinterConnection	AddPrinterConnection
	0x2c	RpcDeletePrinterConnection	DeletePrinterConnection
	0x2d	RpcPrinterMessageBox	
	0x2e	RpcAddMonitor	AddMonitor
	0x2f	RpcDeleteMonitor	DeleteMonitor
	0x30	RpcDeletePrintProcessor	DeletePrintProcessor
	0x31	RpcAddPrintProvidor	AddPrintProvidor
	0x32	RpcDeletePrintProvidor	DeletePrintProvidor
	0x33	RpcEnumPrintProcessorDatatypes	EnumPrintProcessorDatatypes
	0x34	RpcResetPrinter	ResetPrinter
	0x35	RpcGetPrinterDriver2	GetPrinterDriver2
	0x36	RpcClientFindFirstPrinterChangeNotification	FindFirstPrinterChangeNotification
	0x37	RpcFindNextPrinterChangeNotification	FindNextPrinterChangeNotification
	0x38	RpcFindClosePrinterChangeNotification	FindClosePrinterChangeNotification
	0x39	RpcRouterFindFirstPrinterChangeNotificationOld	
	0x3a	RpcReplyOpenPrinter	
	0x3b	RpcRouterReplyPrinter	
	0x3c	RpcReplyClosePrinter	
	0x3d	RpcAddPortEx	
	0x3e	RpcRemoteFindFirstPrinterChangeNotification	
	0x3f	RpcSpoolerInit	
	0x40	RpcResetPrinterEx	
	0x41	RpcRemoteFindFirstPrinterChangeNotificationEx	
	0x42	RpcRouterReplyPrinterEx	
	0x43	RpcRouterRefreshPrinterChangeNotification	
	0x44	RpcSetAllocFailCount	
	0x45	RpcSplOpenPrinter	

	0x46	RpcAddPrinterEx	
	0x47	RpcSetPort	
	0x48	RpcEnumPrinterData	
	0x49	RpcDeletePrinterData	
	0x4a	RpcClusterSplOpen	
	0x4b	RpcClusterSplClose	
	0x4c	RpcClusterSplIsAlive	
	0x4d	RpcSetPrinterDataEx	
	0x4e	RpcGetPrinterDataEx	
	0x4f	RpcEnumPrinterDataEx	
	0x50	RpcEnumPrinterKey	
	0x51	RpcDeletePrinterDataEx	
	0x52	RpcDeletePrinterKey	
	0x53	RpcSeekPrinter	
	0x54	RpcDeletePrinterDriverEx	
	0x55	RpcAddPerMachineConnection	
	0x56	RpcDeletePerMachineConnection	
	0x57	RpcEnumPerMachineConnections	
	0x58	RpcXcvData	
	0x59	RpcAddPrinterDriverEx	
	0x5a	RpcSplOpenPrinter	
	0x5b	RpcGetSpoolFileInfo	
	0x5c	RpcCommitSpoolData	
	0x5d	RpcCloseSpoolFileHandle	
	0x5e	RpcFlushPrinter	FlushPrinter
> Windows XP and Windows Server 2003	0x5f	RpcSendRecvBidiData	
	0x60	RpcAddDriverCatalog	
> Windows Vista	0x61	RpcAddPrinterConnection2	
	0x62	RpcDeletePrinterConnection2	
	0x63	RpcInstallPrinterDriverFromPackage	
	0x64	RpcUploadPrinterDriverPackage	
	0x65	RpcGetCorePrinterDrivers	
	0x66	RpcCorePrinterDriverInstalled	

	0x67	RpcGetPrinterDriverPackagePath	
	0x68	RpcReportJobProcessingProgress	

In August 2005, a security vulnerability discovered by Kostya Kortchinsky was fixed by Microsoft in the MS05-043 security bulletin [67]. The vulnerability can be exploited calling the **AddPrinterEx** operation (opnum 0x46).

Prev | Up | Next

4.10.15. Scheduler service | Home | 4.10.17. WINS service

4.10.17. WINS service

The WINS service (**wins.exe** process) runs two RPC services, available on two endpoints:

- A dynamic TCP port
- **WinsPipe** named pipe

The two RPC services identifiers are:

```
45f52c28-7f9f-101a-b52b-08002b2efabe v1.0
811109bf-a4e1-11d1-ab54-00a0c91e9b45 v1.0
```

Table 4.66. winsif operations

Interface	Operation number	Operation name
45f52c28-7f9f-101a-b52b-08002b2efabe v1.0: winsif		
	0x00	R_WinsRecordAction
	0x01	R_WinsStatus
	0x02	R_WinsTrigger
	0x03	R_WinsDoStaticInit
	0x04	R_WinsDoScavenging
	0x05	R_WinsGetDbRecs
	0x06	R_WinsTerm
	0x07	R_WinsBackup
	0x08	R_WinsDelDbRecs
	0x09	R_WinsPullRange
	0x0a	R_WinsSetPriorityClass
	0x0b	R_WinsResetCounters
	0x0c	R_WinsWorkerThdUpd
	0x0d	R_WinsGetNameAndAdd

	0x0e	R_WinsGetBrowserNames_Old
	0x0f	R_WinsDeleteWins
	0x10	R_WinsSetFlags
	0x11	R_WinsGetDbRecsByName
	0x12	R_WinsStatusWHdl
	0x13	R_WinsDoScavengingNew

Table 4.67. winsi2 operations

Interface	Operation number	Operation name
811109bf-a4e1-11d1-ab54-00a0c91e9b45 v1.0: winsi2		
	0x00	R_WinsTombstoneDbRecs
	0x01	R_WinsCheckAccess

The WINS service also opens a dynamic UDP port, which does not seem to be used by a RPC service.

4.11. Other MSRPC interfaces

4.11.1. Application Management service
4.11.2. Certificate services
4.11.3. Client Service for NetWare
4.11.4. Cryptographic Services service
4.11.5. DHCP Client service
4.11.6. DHCP Server service
4.11.7. Distributed Link Tracking Client service
4.11.8. Distributed Link Tracking Server service
4.11.9. DNS Client service - Windows 2000
4.11.10. DNS Client service - Windows XP and later versions
4.11.11. EFS
4.11.12. Fax server
4.11.13. File Server for Macintosh
4.11.14. IPsec Policy Agent service - Windows 2000
4.11.15. IPsec Services service - Windows XP and later versions
4.11.16. License Logging service
4.11.17. Microsoft SQL Server
4.11.18. Protected storage service
4.11.19. Routing and Remote Access service
4.11.20. Secondary Logon service
4.11.21. Security Configuration Editor Engine
4.11.22. SSDP Discovery Service service
4.11.23. System Event Notification service
4.11.24. Telephony service
4.11.25. Terminal Server service
4.11.26. WebClient service
4.11.27. Windows Audio service
4.11.28. Windows File Protection
4.11.29. Windows Security Center
4.11.30. Windows Time service
4.11.31. Winlogon process - Windows 2000

4.11.32. Winlogon process - Windows Server 2003
4.11.33. Wireless Configuration service

Prev	Up	Next
4.10.17. WINS service	Home	4.11.1. Application Management service

4.11.1. Application Management service

The Application Management service runs one RPC service, available on the following endpoint:

- **appmgmt** LPC port

```
Z:\>ifids -p ncalrpc -e appmgmt serveur
Interfaces: 47

[...]

  8c7daf44-b6dc-11d1-9a4c-0020af6e7c57 v1.0
```

Table 4.68. appmgmt operations

Interface	Operation number	Operation name
8c7daf44-b6dc-11d1-9a4c-0020af6e7c57 v1.0: appmgmt		
	0x00	PINSTALLCONTEXT_rundown
	0x01	InstallBegin
	0x02	InstallManageApp
	0x03	InstallUnmanageApp
	0x04	InstallEnd
	0x05	ARPRemoveApp
	0x06	GetManagedApps
	0x07	RsopReportInstallFailure
	0x08	GetManagedAppCategories

4.11.2. Certificate services

The certificate services runs one RPC service on the following endpoint:

- **cert** named pipe

```
Y:\>ifids -p ncacn_np -e \pipe\cert \\.
Interfaces: 6

[...]

  91ae6020-9e3c-11cf-8d7c-00aa00c091be v0.0

[...]
```

Excerpt of the How Certificate Services Works section of Windows Server 2003 documentation:

> The ICertPassage RPC protocol provides an interface to a CA for submitting a PKCS #10 request and receiving a certificate in a PKCS #7 response. ICertPassage has been superseded in Windows Server 2003 by ICertRequest, but is still used by Windows 2000 clients.

Table 4.69. ICertPassage operations

Interface	Operation number	Operation name
91ae6020-9e3c-11cf-8d7c-00aa00c091be v0.0: ICertPassage		
-	0x00	CertServerRequest

Excerpt of the How Certificate Services Works section of Windows Server 2003 documentation:

Certificate Services publishes the following external COM interfaces:

- ICertRequest and ICertRequest2. These interfaces accept requests for new and existing certificates.
- ICertAdmin and ICertAdmin2. These interfaces provide administrative functions, including

request management (modification, denial, resubmission, and revocation), importing certificates, key archival, CRL management, server configuration, user role retrieval, and certificate data retrieval.

- ICertView and ICertView2. These interfaces allow properly authorized clients to create a customized or complete view of the Certificate Services certificates database.

4.11.3. Client Service for NetWare

Table 4.70. nwwks operations

Interface	Operation number	Operation name
e67ab081-9844-3521-9d32-834f038001c0 v1.0: nwwks		
	0x00	NwrCreateConnection
	0x01	NwrChangePassword
	0x02	NwrDeleteConnection
	0x03	NwrQueryServerResource
	0x04	NwrOpenEnumConnections
	0x05	NwrOpenEnumContextInfo>
	0x06	NwrOpenEnumServersAndNdsTrees
	0x07	NwrOpenEnumNdsSubTrees_Disk
	0x08	NwrOpenEnumNdsSubTrees_Print
	0x09	NwrOpenEnumNdsSubTrees_Any
	0x0a	NwrOpenEnumVolumes
	0x0b	NwrOpenEnumQueues
	0x0c	NwrOpenEnumVolumeQueues
	0x0d	NwrOpenEnumDirectories
	0x0e	NwrEnum
	0x0f	NwrEnumConnections
	0x10	NwrCloseEnum
	0x11	NwrLogonUser
	0x12	NwrLogoffUser
	0x13	NwrSetInfo
	0x14	NwrValidateUser
	0x15	NwrOpenPrinter

	0x16	NwrClosePrinter	
	0x17	NwrGetPrinter	
	0x18	NwrSetPrinter	
	0x19	NwrEnumPrinters	
	0x1a	NwrStartDocPrinter	
	0x1b	NwrWritePrinter	
	0x1c	NwrAbortPrinter	
	0x1d	NwrEndDocPrinter	
	0x1e	NwrEnumJobs	
	0x1f	NwrGetJob	
	0x20	NwrSetJob	
	0x21	NwrAddJob	
	0x22	NwrScheduleJob	
	0x23	NwrWaitForPrinterChange	
	0x24	NwrEnumGWDevices	
	0x25	NwrAddGWDevice	
	0x26	NwrDeleteGWDevice	
	0x27	NwrQueryGatewayAccount	
	0x28	NwrSetGatewayAccount	
	0x29	NwrGetService	
	0x2a	NwrSetService	
	0x2b	NwrGetUser	
	0x2c	NwrGetResourceInformation	
	0x2d	NwrGetConnectionPerformance	
	0x2e	NwrGetResourceParent	
	0x2f	NwrSetLogonScript	

4.11.4. Cryptographic Services service

The Cryptographic Services service runs three RPC services, available on the following endpoints:

- **keysvc** LPC port
- **keysvc** named pipe

```
Y:\>ifids -p ncalrpc -e keysvc serveur
Interfaces: 40

[...]

  8d0ffe72-d252-11d0-bf8f-00c04fd9126b v1.0
  0d72a7d4-6148-11d1-b4aa-00c04fb66ea0 v1.0
  f50aac00-c7f3-428e-a022-a6b71bfb9d43 v.1.0

[...]

Y:\>ifids -p ncacn_np -e \pipe\keysvc \\.
Interfaces: 40

[...]

  8d0ffe72-d252-11d0-bf8f-00c04fd9126b v1.0
  0d72a7d4-6148-11d1-b4aa-00c04fb66ea0 v1.0
  f50aac00-c7f3-428e-a022-a6b71bfb9d43 v.1.0

[...]
```

Table 4.71. IKeySvc operations

Interface	Operation number	Operation name
8d0ffe72-d252-11d0-bf8f-00c04fd9126b v1.0: IKeySvc		
	0x00	KeyrOpenKeyService

	0x01	KeyrEnumerateProviders
	0x02	KeyrEnumerateProviderTypes
	0x03	KeyrEnumerateProvContainers
	0x04	KeyrCloseKeyService
	0x05	KeyrGetDefaultProvider
	0x06	KeyrSetDefaultProvider
	0x07	KeyrEnroll
	0x08	KeyrExportCert
	0x09	KeyrImportCert
	0x0a	KeyrEnumerateAvailableCertTypes
	0x0b	KeyrEnumerateCAs
	0x0c	KeyrEnroll_V2
	0x0d	KeyrQueryRequestStatus

In Windows Vista, the **IKeySvc** interface is replaced by the **IKeySvc2** interface, with the following endpoints:

- **keysvc** LPC port
- **keysvc** named pipe
- **keysvc2** LPC port

```
Y:\>ifids -p ncalrpc -e keysvc2 vista
Interfaces: 56

[...]

  68b58241-c259-4f03-a2e5-a2651dcbc930 v1.0
  0d72a7d4-6148-11d1-b4aa-00c04fb66ea0 v1.0
  f50aac00-c7f3-428e-a022-a6b71bfb9d43 v.1.0

[...]
```

Table 4.72. IKeySvc2 operations

Interface	Operation number	Operation name
68b58241-c259-4f03-a2e5-a2651dcbc930 v1.0: IKeySvc2		

	0x00	KSrSubmitRequest
	0x01	KSrGetTemplates
	0x02	KSrGetCAs

Table 4.73. ICertProtect operations

Interface	Operation number	Operation name
0d72a7d4-6148-11d1-b4aa-00c04fb66ea0 v1.0: ICertProtect		
	0x00	SSCertProtectFunction

Table 4.74. ICatDBSvc operations

Interface	Operation number	Operation name
f50aac00-c7f3-428e-a022-a6b71bfb9d43 v.1.0: ICatDBSvc		
	0x00	SSCatDBAddCatalog
	0x01	SSCatDBDeleteCatalog
	0x02	SSCatDBEnumCatalogs
	0x03	SSCatDBRegisterForChangeNotification
	0x04	KeyrCloseKeyService
	0x04	SSCatDBRebuildDatabase

4.11.5. DHCP Client service

In Windows Server 2003, the DHCP Client service is running in an svchost.exe instance running under the NETWORK SERVICE logon session. The DNS Client service is running in the same process.

The DHCP Client service runs one RPC service, available on the following endpoint:

- **dhcpcsvc** LPC port

```
Y:\>ifids -p ncalrpc -e dhcpcsvc serveur
Interfaces: 2
  3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 v1.0
  45776b01-5956-4485-9f80-f428f7d60129 v2.0
```

In Windows Vista, the following two interfaces are registered in the endpoint mapper database:

```
[...]

IfId: 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 version 1.0
Annotation: DHCP Client LRPC Endpoint
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:127.0.0.1[49154]

IfId: 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d6 version 1.0
Annotation: DHCPv6 Client LRPC Endpoint
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:127.0.0.1[49154]

[...]
```

In Windows Vista, these interfaces are local-only, using the following endpoints:

- **dhcpcsvc** LPC port
- **dhcpcsvc6** LPC port

Table 4.75. RpcSrvDHCPC operations

Interface	Operation number	Operation name
3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 v1.0: RpcSrvDHCPC		
	0x00	RpcSrvEnableDhcp
	0x01	RpcSrvRenewLease
	0x02	RpcSrvRenewLeaseByBroadcast
	0x03	RpcSrvReleaseLease
	0x04	RpcSrvSetFallbackParams
	0x05	RpcSrvGetFallbackParams
	0x06	RpcSrvFallbackRefreshParams
	0x07	RpcSrvStaticRefreshParams
	0x08	RpcSrvRemoveDnsRegistrations
	0x09	RpcSrvRequestParams
	0x0a	RpcSrvPersistentRequestParams
	0x0b	RpcSrvRegisterParams
	0x0c	RpcSrvDeRegisterParams
	0x0d	RpcSrvEnumInterfaces
	0x0e	RpcSrvQueryLeaseInfo
	0x0f	RpcSrvSetClassId
	0x10	RpcSrvGetClassId
	0x11	RpcSrvSetClientId
	0x12	RpcSrvGetClientId
	0x13	RpcSrvNotifyMediaReconnected
	0x14	RpcSrvGetOriginalSubnetMask
	0x15	RpcSrvSetMSFTVendorSpecificOptions
	0x16	RpcSrvRequestCachedParams
	0x17	RpcSrvRegisterConnectionStateNotification
	0x18	RpcSrvDeRegisterConnectionStateNotification
	0x19	RpcSrvGetNotificationStatus

	0x1a	RpcSrvGetDhcpServicedConnections

Table 4.76. dhcpcsvc6 operations

Interface	Operation number	Operation name
3c4728c5-f0ab-448b-bda1-6ce01eb0a6d6 version 1.0: dhcpcsvc6		
	0x00	RpcSrvRequestPrefix
	0x01	RpcSrvRenewPrefix
	0x02	RpcSrvReleasePrefix
	0x03	RpcSrvRequestParams

Prev Up Next
4.11.4. Cryptographic Services service Home 4.11.6. DHCP Server service

4.11.6. DHCP Server service

The DHCP Server service runs two RPC services, available on the following endpoint:

- **DHCPSERVERLPC** LPC port

```
Z:\>ifids -p ncalrpc -e DHCPSERVERLPC serveur
Interfaces: 6
  00000134-0000-0000-c000-000000000046 v0.0
  18f70770-8e64-11cf-9af1-0020af6e72f4 v0.0
  00000131-0000-0000-c000-000000000046 v0.0
  00000143-0000-0000-c000-000000000046 v0.0
  6bffd098-a112-3610-9833-46c3f874532d v1.0
  5b821720-f63b-11d0-aad2-00c04fc324db v1.0
```

Table 4.77. dhcpsrv operations

Interface	Operation number	Operation name
6bffd098-a112-3610-9833-46c3f874532d v1.0: dhcpsrv		
	0x00	R_DhcpCreateSubnet
	0x01	R_DhcpSetSubnetInfo
	0x02	R_DhcpGetSubnetInfo
	0x03	R_DhcpEnumSubnets
	0x04	R_DhcpAddSubnetElement
	0x05	R_DhcpEnumSubnetElements
	0x06	R_DhcpRemoveSubnetElement
	0x07	R_DhcpDeleteSubnet
	0x08	R_DhcpCreateOption
	0x09	R_DhcpSetOptionInfo
	0x0a	R_DhcpGetOptionInfo
	0x0b	R_DhcpRemoveOption

	0x0c	R_DhcpSetOptionValue
	0x0d	R_DhcpGetOptionValue
	0x0e	R_DhcpEnumOptionValues
	0x0f	R_DhcpRemoveOptionValue
	0x10	R_DhcpCreateClientInfo
	0x11	R_DhcpSetClientInfo
	0x12	R_DhcpGetClientInfo
	0x13	R_DhcpDeleteClientInfo
	0x14	R_DhcpEnumSubnetClients
	0x15	R_DhcpGetClientOptions
	0x16	R_DhcpGetMibInfo
	0x17	R_DhcpEnumOptions
	0x18	R_DhcpSetOptionValues
	0x19	R_DhcpServerSetConfig
	0x1a	R_DhcpServerGetConfig
	0x1b	R_DhcpScanDatabase
	0x1c	R_DhcpGetVersion
	0x1d	R_DhcpAddSubnetElementV4
	0x1e	R_DhcpEnumSubnetElementsV4
	0x1f	R_DhcpRemoveSubnetElementV4
	0x20	R_DhcpCreateClientInfoV4
	0x21	R_DhcpSetClientInfoV4
	0x22	R_DhcpGetClientInfoV4
	0x23	R_DhcpEnumSubnetClientsV4
	0x24	R_DhcpSetSuperScopeV4
	0x25	R_DhcpGetSuperScopeInfoV4
	0x26	R_DhcpDeleteSuperScopeV4
	0x27	R_DhcpServerSetConfigV4
	0x28	R_DhcpServerGetConfigV4

Table 4.78. dhcpsrv2 operations

Interface	Operation number	Operation name
5b821720-f63b-11d0-aad2-00c04fc324db v1.0: dhcpsrv2		
-	0x00	R_DhcpEnumSubnetClientsV5
-	0x01	R_DhcpSetMScopeInfo
-	0x02	R_DhcpGetMScopeInfo
-	0x03	R_DhcpEnumMScopes
-	0x04	R_DhcpAddMScopeElement
-	0x05	R_DhcpEnumMScopeElements
-	0x06	R_DhcpRemoveMScopeElement
-	0x07	R_DhcpDeleteMScope
-	0x08	R_DhcpScanMDatabase
-	0x09	R_DhcpCreateMClientInfo
-	0x0a	R_DhcpSetMClientInfo
-	0x0b	R_DhcpGetMClientInfo
-	0x0c	R_DhcpDeleteMClientInfo
-	0x0d	R_DhcpEnumMScopeClients
-	0x0e	R_DhcpCreateOptionV5
-	0x0f	R_DhcpSetOptionInfoV5
-	0x10	R_DhcpGetOptionInfoV5
-	0x11	R_DhcpEnumOptionsV5
-	0x12	R_DhcpRemoveOptionV5
-	0x13	R_DhcpSetOptionValueV5
-	0x14	R_DhcpSetOptionValuesV5
-	0x15	R_DhcpGetOptionValueV5
-	0x16	R_DhcpEnumOptionValuesV5
-	0x17	R_DhcpRemoveOptionValueV5
-	0x18	R_DhcpCreateClass
-	0x19	R_DhcpModifyClass
-	0x1a	R_DhcpDeleteClass
-	0x1b	R_DhcpGetClassInfo
-	0x1c	R_DhcpEnumClasses

-	0x1d	R_DhcpGetAllOptions
-	0x1e	R_DhcpGetAllOptionValues
-	0x1f	R_DhcpGetMCastMibInfo
-	0x20	R_DhcpAuditLogSetParams
-	0x21	R_DhcpAuditLogGetParams
-	0x22	R_DhcpServerQueryAttribute
-	0x23	R_DhcpServerQueryAttributes
-	0x24	R_DhcpServerRedoAuthorization
-	0x25	R_DhcpAddSubnetElementV5
-	0x26	R_DhcpEnumSubnetElementsV5
-	0x27	R_DhcpRemoveSubnetElementV5
-	0x28	R_DhcpGetServerBindingInfo
-	0x29	R_DhcpSetServerBindingInfo
-	0x2a	R_DhcpQueryDnsRegCredentials
-	0x2b	R_DhcpSetDnsRegCredentials
-	0x2c	R_DhcpBackupDatabase
-	0x2d	R_DhcpRestoreDatabase

Prev | Up | Next

4.11.5. DHCP Client service | Home | 4.11.7. Distributed Link Tracking Client service

4.11.7. Distributed Link Tracking Client service

The Distributed Link Tracking Client service, implemented in the **trkwks.dll** DLL, runs one RPC service, available on the following endpoints:

- **trkwks** LPC port
- **trkwks** named pipe

```
Y:\>ifids -p ncalrpc -e trkwks serveur
Interfaces: 40

[...]

  300f3532-38cc-11d0-a3f0-0020af6b0add v1.2

Y:\>ifids -p ncacn_np -e \pipe\trkwks \\.
Interfaces: 40

[...]

  300f3532-38cc-11d0-a3f0-0020af6b0add v1.2
```

Table 4.79. trkwks operations

Interface	Operation number	Operation name
300f3532-38cc-11d0-a3f0-0020af6b0add v1.2: trkwks		
	0x00	LnkMendLink
	0x01	LnkSearchMachine
	0x02	LnkCallSvrMessage
	0x03	LnkSetVolumeId
	0x04	LnkRestartDcSynchronization
	0x05	GetVolumeTrackingInformation

	0x06	GetFileTrackingInformation
	0x07	TriggerVolumeClaims
	0x08	LnkOnRestore
	0x09	LnkMendLink
	0x0a	LnkSearchMachine
	0x0b	LnkCallSvrMessage
	0x0c	LnkSearchMachine

Prev | Up | Next

4.11.6. DHCP Server service | Home | 4.11.8. Distributed Link Tracking Server service

4.11.8. Distributed Link Tracking Server service

Table 4.80. trksvr operations

Interface	Operation number	Operation name
4da1c422-943d-11d1-acae-00c04fc2aa3f v1.0: trksvr		
	0x00	LnkSvrMessage

4.11.9. DNS Client service - Windows 2000

0n Windows 2000, the DNS Client service (caching DNS resolver) runs one RPC service.

Table 4.81. dnsrslvr operations

Interface	Operation number	Operation name
65a93890-fab9-43a3-b2a5-1e330ac28f11 v2.0: dnsrslvr		
	0x00	CRrFlushCache
	0x01	CRrFlushCacheEntry
	0x02	CRrFlushCacheEntryForType
	0x03	CRrTrimCache
	0x04	CRrReadCache
	0x05	CRrReadCacheEntry
	0x06	CRrQuery
	0x07	CRrGetAdapterInfo
	0x08	CRrGetSearchList
	0x09	CRrGetPrimaryDomainName
	0x0a	CRrGetIpAddressList
	0x0b	CRrGetHashTableStats
	0x0c	CRrRegisterParamChange
	0x0d	CRrDeregisterParamChange
	0x0e	CRrUpdateTest
	0x0f	CRrCacheRecordSet

4.11.10. DNS Client service - Windows XP and later versions

Starting with Windows XP, the DNS Client service runs one RPC service, available on the following endpoint:

- **DNSResolver** LPC port

```
Y:\>ifids -p ncalrpc -e DNSResolver serveur
Interfaces: 1
  45776b01-5956-4485-9f80-f428f7d60129 v2.0
```

Table 4.82. DnsResolver operations

Interface	Operation number	Operation name
45776b01-5956-4485-9f80-f428f7d60129 v2.0: DnsResolver		
	0x00	CRrReadCache
	0x01	CRrReadCacheEntry
	0x02	CRrGetHashTableStats
	0x03	R_ResolverGetConfig
	0x04	R_ResolverFlushCache
	0x05	R_ResolverFlushCacheEntry
	0x06	R_ResolverRegisterCluster
	0x07	R_ResolverQuery
	0x08	R_ResolverEnumCache
	0x09	R_ResolverPoke - R_ResolverSimpleOp

4.11.11. EFS

The EFS (Encrypted File System) subsystem runs one RPC service, **efsrpc**, used to communicate with the service that implement cryptographic operations on the local system.

Table 4.83. efsrpc operations

Interface	Operation number	Operation name
c681d488-d850-11d0-8c52-00c04fd90f7e v1.0: efsrpc		
	0x00	EfsRpcOpenFileRaw
	0x01	EfsRpcReadFileRaw
	0x02	EfsRpcWriteFileRaw
	0x03	EfsRpcCloseRaw
	0x04	EfsRpcEncryptFileSrv
	0x05	EfsRpcDecryptFileSrv
	0x06	EfsRpcQueryUserOnFile
	0x07	EfsRpcQueryRecoveryAgents
	0x08	EfsRpcRemoveUsersFromFile
	0x09	EfsRpcAddUsersToFile
	0x0a	EfsRpcSetFileEncryptionKey
	0x0b	EfsRpcNotSupported
> Windows XP and Windows Server 2003	0x0c	EfsRpcFileKeyInfo
	0x0d	EfsRpcDuplicateEncryptionInfoFile
> Windows Vista	0x0e	EfsUsePinForEncryptedFiles
	0x0f	EfsRpcEncryptFileCse
	0x10	EfsRpcDecryptFileSrvEx
	0x11	EfsRpcRemoveUsersFromFileEx
	0x12	EfsRpcAddUsersToFileEx

	0x13	EfsRpcDuplicateEncryptionInfoFileEx
	0x14	EfsRpcFileKeyInfoEx
	0x15	EfsRpcGenerateEfsStream
	0x16	EfsRpcGetEncryptedFileMetadata
	0x17	EfsRpcSetEncryptedFileMetadata
	0x18	EfsRpcFlushEfsCache

Prev | Up | Next

4.11.10. DNS Client service - Windows XP and later versions

Home

4.11.12. Fax server

4.11.12. Fax server

Table 4.84. fax_Server operations

Interface	Operation number	Operation name	Windows API
ea0a3165-4834-11d2-a6f8-00c04fa346cc v4.0: fax_Server			
	0x00	GetServicePrinters	
	0x01	ConnectionRefCount	
	0x02	OpenPort	
	0x03	ClosePort	
	0x04	EnumJobs	
	0x05	GetJobs>	
	0x06	SetJobs	
	0x07	GetPageData	
	0x08	GetDeviceStatus	
	0x09	Abort	
	0x0a	EnumPorts	
	0x0b	GetPort	
	0x0c	SetPort	
	0x0d	EnumRoutingMethods	
	0x0e	EnableRoutingMethod	
	0x0f	GetRoutingInfo	
	0x10	SetRoutingInfo	
	0x11	EnumGlobalRoutingInfo	
	0x12	SetGlobalRoutingInfo	
	0x13	GetConfiguration	
	0x14	SetConfiguration	

	0x15	GetLoggingCategories	
	0x16	SetLoggingCategories	
	0x17	GetSecurity	
	0x18	SetSecurity	
	0x19	AccessCheck	
	0x1a	CheckServerProtSeq	
	0x1b	SendDocumentEx	
	0x1c	EnumJobsEx	
	0x1d	GetJobEx	
	0x1e	GetCountryList	
	0x1f	GetPersonalProfileInfo	
	0x20	GetQueueStates	
	0x21	SetQueue	
	0x22	GetReceiptsConfiguration	
	0x23	SetReceiptsConfiguration	
	0x24	GetReceiptsOptions	
	0x25	GetVersion	
	0x26	GetOutboxConfiguration	
	0x27	SetOutboxConfiguration	
	0x28	GetPersonalCoverPagesOption	
	0x29	GetArchiveConfiguration	
	0x2a	SetArchiveConfiguration	
	0x2b	GetActivityLoggingConfiguration	
	0x2c	SetActivityLoggingConfiguration	
	0x2d	EnumerateProviders	
	0x2e	GetPortEx	
	0x2f	SetPortEx	
	0x30	EnumPortsEx	
	0x31	GetExtensionData	
	0x32	SetExtensionData	
	0x33	AddOutboundGroup	
	0x34	SetOutboundGroup	

	0x35	RemoveOutboundGroup	
	0x36	EnumOutboundGroups	
	0x37	SetDeviceOrderInGroup	
	0x38	AddOutboundRule	
	0x39	RemoveOutboundRule	
	0x3a	SetOutboundRule	
	0x3b	EnumOutboundRules	
	0x3c	RegisterServiceProviderEx	
	0x3d	UnregisterServiceProviderEx	
	0x3e	UnregisterRoutingExtension	
	0x3f	StartMessagesEnum	
	0x40	EndMessagesEnum	
	0x41	EnumMessages	
	0x42	GetMessage	
	0x43	RemoveMessage	
	0x44	StartCopyToServer	
	0x45	StartCopyMessageFromServer	
	0x46	WriteFile	
	0x47	ReadFile	
	0x48	EndCopy	
	0x49	StartServerNotification	
	0x4a	StartServerNotificationEx	
	0x4b	EndServerNotification	
	0x4c	GetServerActivity	
	0x4d	SetConfigWizardUsed	
	0x4e	EnumRoutingExtensions	
	0x4f	AnswerCall	
	0x50	ConnectFaxServer	
	0x51	GetSecurityEx	
	0x52	RefreshArchive	
	0x53	SetRecipientsLimit	
	0x54	GetRecipientsLimit	

	0x55	GetServerSKU	
	0x56	CheckValidFaxFolder	
	0x57	GetJobEx2	
	0x58	EnumJobsEx2	
	0x59	GetMessageEx	
	0x5a	StartMessageEnumEx	
	0x5b	EnumMessagesEx	
	0x5c	StartServerNotificationEx2	
	0x5d	CreateAccount	
	0x5e	DeleteAccount	
> Windows XP and Windows Server 2003	0x5f	EnumAccounts	
	0x60	GetAccountInfo	
> Windows Vista	0x61	GetGeneralConfiguration	
	0x62	SetGeneralConfiguration	
	0x63	GetSecurityEx2	
	0x64	SetSecurityEx2	
	0x65	AccessCheckEx2	
	0x66	ReAssignMessage	
	0x67	SetMessage	
	0x68	GetConfigOption	

4.11.13. File Server for Macintosh

Table 4.85. sfmsvc operations

Interface	Operation number	Operation name
4b324fc8-1670-01d3-1278-5a47bf6ee188 v0.0: sfmsvc		
	0x00	AfpAdminrVolumeEnum
	0x01	AfpAdminrVolumeSetInfo
	0x02	AfpAdminrVolumeGetInfo
	0x03	AfpAdminrVolumeDelete
	0x04	AfpAdminrVolumeAdd
	0x05	AfpAdminrInvalidVolumeEnum
	0x06	AfpAdminrInvalidVolumeDelete
	0x07	AfpAdminrDirectoryGetInfo
	0x08	AfpAdminrDirectorySetInfo
	0x09	AfpAdminrServerGetInfo
	0x0a	AfpAdminrServerSetInfo
	0x0b	AfpAdminrSessionEnum
	0x0c	AfpAdminrSessionClose
	0x0d	AfpAdminrConnectionEnum
	0x0e	AfpAdminrConnectionClose
	0x0f	AfpAdminrFileEnum
	0x10	AfpAdminrFileClose
	0x11	AfpAdminrETCMapGetInfo
	0x12	AfpAdminrETCMapAdd
	0x13	AfpAdminrETCMapDelete
	0x14	AfpAdminrETCMapSetInfo
	0x15	AfpAdminrETCMapAssociate

	0x16	AfpAdminrStatisticsGet
	0x17	AfpAdminrStatisticsGetEx
	0x18	AfpAdminrStatisticsClear
	0x19	AfpAdminrProfileGet
	0x1a	AfpAdminrProfileClear
	0x1b	AfpAdminrMessageSend
	0x1c	AfpAdminrFinderSetInfo

Prev | Up | Next

4.11.12. Fax server | Home | 4.11.14. IPsec Policy Agent service - Windows 2000

4.11.14. IPsec Policy Agent service - Windows 2000

On Windows 2000, the IPsec Policy Agent service runs one RPC service, available on the following endpoints:

- **policyagent** LPC port
- **POLICYAGENT** named pipe

```
C:\>ifids -p ncalrpc -e policyagent fenetre
Interfaces: 5

[...]

  d335b8f6-cb31-11d0-b0f9-006097ba4e54 v1.5

C:\>ifids -p ncacn_np -e \pipe\policyagent \\.
Interfaces: 5

[...]

  d335b8f6-cb31-11d0-b0f9-006097ba4e54 v1.5
```

Table 4.86. PolicyAgent operations

Interface	Operation number	Operation name
d335b8f6-cb31-11d0-b0f9-006097ba4e54 v1.5: PolicyAgent		
	0x00	PAAddPolicyRule
	0x01	PAUpdatePolicyRule
	0x02	PADeletePolicy
	0x03	PAQueryIsakmpPolicy
	0x04	PAAddIsakmpPolicy
	0x05	PARefreshPolicies
	0x06	PAAddFilter

	0x07	PAMatchFilter
	0x08	PAQueryIpsecPolicy
	0x09	PAQueryFilters
	0x0a	PADeleteFilter
	0x0b	PAQueryStatistics
	0x0c	PAQueryAssociations
	0x0d	PAQueryIsakmpAssociations
	0x0e	PADeleteIsakmpAssociation
	0x0f	IsakmpInitiateNegotiation
	0x10	IsakmpQueryNegotiationStatus
	0x11	IsakmpCloseNegotiationStatusHandle
	0x12	IsakmpQuerySpiChange
	0x13	IsakmpRegisterNotifyClient
	0x14	IsakmpDeregisterNotifyClient
	0x15	IsakmpQuerySpi

Prev | Up | Next

4.11.13. File Server for Macintosh | Home | 4.11.15. IPsec Services service - Windows XP and later versions

4.11.15. IPsec Services service - Windows XP and later versions

In Windows XP, the IPsec Services service runs one RPC service on the following endpoints:

- **ipsec** LPC port
- **ipsec** named pipe

```
E:\>ifids -p ncalrpc -e ipsec jamal
Interfaces: 8

[...]

  12345678-1234-abcd-ef00-0123456789ab v1.0


E:\>ifids -p ncacn_np -e \pipe\ipsec \\.

Interfaces: 8

[...]

  12345678-1234-abcd-ef00-0123456789ab v1.0
```

In Windows Server 2003, the RPC service does not seem to set a specific endpoint. If the **HKLM \SYSTEM\CCS\Services\PolicyAgent\EnableRemoteMgmt** registry value is set to 0 or is not present, the RPC security callback function prevents remote access to this interface.

In Windows Vista, if the **EnableRemoteMgmt** registry value is set (it is not set by default), the IPsec service registers a named pipe endpoint with a randomly-generated name:

```
C:\> rpcdump 127.0.0.1

[...]

IfId: 12345678-1234-abcd-ef00-0123456789ab v1.0
```

```
Annotation: IPSec Policy Agent endpoint
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:127.0.0.1[\\pipe\\d58b3ca461625de0]

[...]

C:\>ifids -p ncacn_np -e \pipe\d58b3ca461625de0 \\.
Interfaces: 1
  12345678-1234-abcd-ef00-0123456789ab v1.0
```

Table 4.87. winipsec operations

Interface	Operation number	Operation name
12345678-1234-abcd-ef00-0123456789ab v1.0: winipsec		
	0x00	RpcAddTransportFilter
	0x01	RpcDeleteTransportFilter
	0x02	RpcEnumTransportFilters
	0x03	RpcSetTransportFilter
	0x04	RpcGetTransportFilter
	0x05	RpcAddQMPolicy
	0x06	RpcDeleteQMPolicy
	0x07	RpcEnumQMPolicies
	0x08	RpcSetQMPolicy
	0x09	RpcGetQMPolicy
	0x0a	RpcAddMMPolicy
	0x0b	RpcDeleteMMPolicy
	0x0c	RpcEnumMMPolicies
	0x0d	RpcSetMMPolicy
	0x0e	RpcGetMMPolicy
	0x0f	RpcAddMMFilter
	0x10	RpcDeleteMMFilter
	0x11	RpcEnumMMFilters
	0x12	RpcSetMMFilter

	0x13	RpcGetMMFilter
	0x14	RpcMatchMMFilter
	0x15	RpcMatchTransportFilter
	0x16	RpcGetQMPolicyByID
	0x17	RpcGetMMPolicyByID
	0x18	RpcAddMMAuthMethods
	0x19	RpcDeleteMMAuthMethods
	0x1a	RpcEnumMMAuthMethods
	0x1b	RpcSetMMAuthMethods
	0x1c	RpcGetMMAuthMethods
	0x1d	RpcInitiateIKENegotiation
	0x1e	RpcQueryIKENegotiationStatus
	0x1f	RpcCloseIKENegotiationHandle
	0x20	RpcEnumMMSAs
	0x21	RpcDeleteMMSAs
	0x22	RpcDeleteQMSAs
	0x23	RpcQueryIKEStatistics
	0x24	RpcRegisterIKENotifyClient
	0x25	RpcQueryIKENotifyData
	0x26	RpcCloseIKENotifyHandle
	0x27	RpcQueryIPSecStatistics
	0x28	RpcEnumQMSAs
	0x29	RpcAddTunnelFilter
	0x2a	RpcDeleteTunnelFilter
	0x2b	RpcEnumTunnelFilters
	0x2c	RpcSetTunnelFilter
	0x2d	RpcGetTunnelFilter
	0x2e	RpcMatchTunnelFilter
	0x2f	RpcOpenMMFilterHandle
	0x30	RpcCloseMMFilterHandle
	0x31	RpcOpenTransportFilterHandle
	0x32	RpcCloseTransportFilterHandle

	0x33	RpcOpenTransportFilterHandle
	0x34	RpcCloseTransportFilterHandle
	0x35	RpcOpenTunnelFilterHandle
	0x36	RpcCloseTunnelFilterHandle
	0x37	RpcEnumIpsecInterfaces
	0x38	RpcAddSAs
	0x39	RpcSetConfigurationVariables
	0x3a	RpcGetConfigurationVariables
	0x3b	RpcQuerySpdPolicyState
> Windows Vista	0x3c	RpcAddMMFilterEx
	0x3d	RpcEnumMMFiltersEx
	0x3e	RpcSetMMFilterEx
	0x3f	RpcGetMMFilterEx
	0x40	RpcMatchMMFilterEx
	0x41	RpcOpenMMFilterHandleEx
	0x42	RpcAddTransportFilterEx
	0x43	RpcEnumTransportFiltersEx
	0x44	RpcSetTransportFilterEx
	0x45	RpcGetTransportFilterEx
	0x46	RpcMatchTransportFilterEx
	0x47	RpcOpenTransportFilterHandleEx
	0x48	RpcQueryRemoteFWRunning

Prev | Up | Next

4.11.14. IPsec Policy Agent service - Windows 2000 | Home | 4.11.16. License Logging service

4.11.16. License Logging service

The License Logging service runs two RPC services, available on the following endpoints:

- **llslpc** LPC port
- **llsrpc** named pipe

```
Y:\>ifids -p ncalrpc -e llslpc serveur
Interfaces: 2
  342cfd40-3c6c-11ce-a893-08002b2e9c6d v0.0
  57674cd0-5200-11ce-a897-08002b2e9c6d v1.0

Y:\>ifids -p ncacn_np -e \pipe\llsrpc \\.
Interfaces: 2
  342cfd40-3c6c-11ce-a893-08002b2e9c6d v0.0
  57674cd0-5200-11ce-a897-08002b2e9c6d v1.0
```

Table 4.88. lls_license operations

Interface	Operation number	Operation name
57674cd0-5200-11ce-a897-08002b2e9c6d v1.0		
	0x00	LlsrLicenseRequestW
	0x01	LlsrLicenseFree

Table 4.89. llsrpc operations

Interface	Operation number	Operation name
342cfd40-3c6c-11ce-a893-08002b2e9c6d v0.0		
	0x00	LlsrConnect
	0x01	LlsrClose
	0x02	LlsrLicenseEnumW
	0x03	LlsrLicenseEnumA

	0x04	LlsrLicenseAddW
	0x05	LlsrLicenseAddA
	0x06	LlsrProductEnumW
	0x07	LlsrProductEnumA
	0x08	LlsrProductAddW
	0x09	LlsrProductAddA
	0x0a	LlsrProductUserEnumW
	0x0b	LlsrProductUserEnumA
	0x0c	LlsrProductServerEnumW
	0x0d	LlsrProductServerEnumA
	0x0e	LlsrProductLicenseEnumW
	0x0f	LlsrProductLicenseEnumA
	0x10	LlsrUserEnumW
	0x11	LlsrUserEnumA
	0x12	LlsrUserInfoGetW
	0x13	LlsrUserInfoGetA
	0x14	LlsrUserInfoSetW
	0x15	LlsrUserInfoSetA
	0x16	LlsrUserDeleteW
	0x17	LlsrUserDeleteA
	0x18	LlsrUserProductEnumW
	0x19	LlsrUserProductEnumA
	0x1a	LlsrUserProductDeleteW
	0x1b	LlsrUserProductDeleteA
	0x1c	LlsrMappingEnumW
	0x1d	LlsrMappingEnumA
	0x1e	LlsrMappingInfoGetW
	0x1f	LlsrMappingInfoGetA
	0x20	LlsrMappingInfoSetW
	0x21	LlsrMappingInfoSetA
	0x22	LlsrMappingUserEnumW
	0x23	LlsrMappingUserEnumA

	0x24	LlsrMappingUserAddW
	0x25	LlsrMappingUserAddA
	0x26	LlsrMappingUserDeleteW
	0x27	LlsrMappingUserDeleteA
	0x28	LlsrMappingAddW
	0x29	LlsrMappingAddA
	0x2a	LlsrMappingDeleteW
	0x2b	LlsrMappingDeleteA
	0x2c	LlsrServerEnumW
	0x2d	LlsrServerEnumA
	0x2e	LlsrServerProductEnumW
	0x2f	LlsrServerProductEnumA
	0x30	LlsrLocalProductEnumW
	0x31	LlsrLocalProductEnumA
	0x32	LlsrLocalProductInfoGetW
	0x33	LlsrLocalProductInfoGetA
	0x34	LlsrLocalProductInfoSetW
	0x35	LlsrLocalProductInfoSetA
	0x36	LlsrServiceInfoGetW
	0x37	LlsrServiceInfoGetA
	0x38	LlsrServiceInfoSetW
	0x39	LlsrServiceInfoSetA
	0x3a	LlsrReplConnect
	0x3b	LlsrReplClose
	0x3c	LlsrReplicationRequestW
	0x3d	LlsrReplicationServerAddW
	0x3e	LlsrReplicationServerServiceAddW
	0x3f	LlsrReplicationServiceAddW
	0x40	LlsrReplicationUserAddW
	0x41	LlsrProductSecurityGetW
	0x42	LlsrProductSecurityGetA
	0x43	LlsrProductSecuritySetW

	0x44	LlsrProductSecuritySetA
	0x45	LlsrProductLicensesGetA
	0x46	LlsrProductLicensesGetW
	0x47	LlsrCertificateClaimEnumA
	0x48	LlsrCertificateClaimEnumW
	0x49	LlsrCertificateClaimAddCheckA
	0x4a	LlsrCertificateClaimAddCheckW
	0x4b	LlsrCertificateClaimAddA
	0x4c	LlsrCertificateClaimAddW
	0x4d	LlsrReplicationCertDbAddW
	0x4e	LlsrReplicationProductSecurityAddW
	0x4f	LlsrReplicationUserAddExW
	0x50	LlsrCapabilityGet
	0x51	LlsrLocalServiceEnumW
	0x52	LlsrLocalServiceEnumA
	0x53	LlsrLocalServiceAddA
	0x54	LlsrLocalServiceAddW
	0x55	LlsrLocalServiceInfoSetW
	0x56	LlsrLocalServiceInfoSetA
	0x57	LlsrLocalServiceInfoGetW
	0x58	LlsrLocalServiceInfoGetA
	0x59	LlsrCloseEx

A vulnerability in one of the llsrpc interface operations was discovered by Kostya Kortchinsky and fixed by the MS05-010 security bulletin [46] in February 2005.

This vulnerability can be exploited anonymously in Windows NT 4.0 and Windows 2000 SP3 because it is possible to connect anonymously to the **llsrpc** named pipe on these systems.

4.11.17. Microsoft SQL Server

In addition to the TDS network protocol (by default, on TCP port 1433), it is possible to query an SQL Server using the RPCnetlib interface. The RPC service is only available if the Multiprotocol transport has been configured using the SQL Server Network utility.

Protocol sequences for the RPCnetlib interface can be configured in the RPC Protocols entry in the Multiprotocol properties. In the following example, three transports were configured: **ncalrpc**, **ncacn_np** and **ncacn_ip_tcp**. The endpoints for each protocol sequence can be obtained by a query to the portmapper.

```
Z:\>rpcdump 127.0.0.1

[...]

IfId: 3f99b900-4d87-101b-99b7-aa0004007f07 version 1.0
Annotation: MS SQL Server
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[LRPC00000d70.00000001]

IfId: 3f99b900-4d87-101b-99b7-aa0004007f07 version 1.0
Annotation: MS SQL Server
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\SERVEUR[\\pipe\\29F8A2BE40935DE9]

IfId: 3f99b900-4d87-101b-99b7-aa0004007f07 version 1.0
Annotation: MS SQL Server
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:127.0.0.1[4103]

[...]

Z:\>ifids -p ncalrpc -e LRPC00000d70.00000001 serveur
Interfaces: 1
  3f99b900-4d87-101b-99b7-aa0004007f07 v1.0

Z:\>ifids -p ncacn_np -e \pipe\29F8A2BE40935DE9 \\.
Interfaces: 1
```

```
  3f99b900-4d87-101b-99b7-aa0004007f07 v1.0

Z:\>ifids -p ncacn_ip_tcp -e 4103 127.0.0.1
Interfaces: 1
  3f99b900-4d87-101b-99b7-aa0004007f07 v1.0
```

Table 4.90. RPCnetlib operations

Interface	Operation number	Operation name
3f99b900-4d87-101b-99b7-aa0004007f07 v1.0: RPCnetlib		
	0x00	RPCopen
	0x01	RPCread
	0x02	RPCwrite
	0x03	RPCtransact
	0x04	RpcCheckForData
	0x05	RPCclose

4.11.18. Protected storage service

The Protected Storage service runs one RPC service, available on the following endpoints:

- **protected_storage** LPC port
- **protected_storage** named pipe

```
Y:\>ifids -p ncalrpc -e protected_storage serveur
Interfaces: 18

[...]

  c9378ff1-16f7-11d0-a0b2-00aa0061426a v1.0

[...]


Y:\>ifids -p ncacn_np -e \pipe\protected_storage \\.
Interfaces: 18

[...]

  c9378ff1-16f7-11d0-a0b2-00aa0061426a v1.0

[...]
```

Table 4.91. IPStoreProv operations

Interface	Operation number	Operation name
c9378ff1-16f7-11d0-a0b2-00aa0061426a v1.0: IPStoreProv		
	0x00	SSPStoreEnumProviders
	0x01	SSGetProvInfo
	0x02	SSGetProvParam

	0x03	SSetProvParam
	0x04	SSAcquireContext
	0x05	SSReleaseContext
	0x06	SSPasswordInterface
	0x07	SSEnumTypes
	0x08	SSEnumSubtypes
	0x09	SSEnumItems
	0x0a	SSGetTypeInfo
	0x0b	SSGetSubtypeInfo
	0x0c	SSCreateType
	0x0d	SSCreateSubtype
	0x0e	SSDeleteType
	0x0f	SSDeleteSubtype
	0x10	SSDeleteItem
	0x11	SSReadItem
	0x12	SSWriteItem
	0x13	SSOpenItem
	0x14	SSCloseItem
	0x15	SSReadAccessRuleset
	0x16	SSWriteAccessRuleset

In Windows Vista, the following RPC interfaces run in the LSA:

```
Y:\>ifids -p ncalrpc -e lsarpc vista
Interfaces: 7

[...]

  11220835-5b26-4d94-ae86-c3e475a809de v1.0
  5cbe92cb-f4be-45c9-9fc9-33e73e557b20 v1.0
  c9ac6db5-82b7-4e55-ae8a-e464ed7b4277 v1.0

[...]
```

```
[...]

IfId: c9ac6db5-82b7-4e55-ae8a-e464ed7b4277 version 1.0
Annotation: Impl friendly name
UUID: 00736665-0000-0000-0000-000000000000
Binding: ncacn_np:127.0.0.1[\\pipe\\lsass]

IfId: c9ac6db5-82b7-4e55-ae8a-e464ed7b4277 version 1.0
Annotation: Impl friendly name
UUID: 00736665-0000-0000-0000-000000000000
Binding: ncacn_np:127.0.0.1[\\PIPE\\protected_storage]

[...]
```

Table 4.92. ICryptProtect operations

Interface	Operation number	Operation name
11220835-5b26-4d94-ae86-c3e475a809de v1.0: ICryptProtect		
	0x00	SSCryptProtectData
	0x01	SSCryptUnprotectData
> Windows Vista	0x02	SSCryptUpdateProtectedState

Table 4.93. PasswordRecovery operations

Interface	Operation number	Operation name
5cbe92cb-f4be-45c9-9fc9-33e73e557b20 v1.0: PasswordRecovery		
	0x00	SSRecoveryQueryStatus
	0x01	SSRecoveryImportRecoveryKey
	0x02	SSRecoverPassword

Table 4.94. BackupKey operations

Interface	Operation number	Operation name

3dde7c30-165d-11d1-ab8f-00805f14db40 v1.0: BackupKey		
	0x00	BackuprKey

4.11.19. Routing and Remote Access service

The Routing and Remote Access service runs one RPC service, available on the following endpoint:

- **ROUTER** named pipe

This interface allows remote administration of network parameters, using the **netsh** command.

```
Y:\>ifids -p ncacn_np -e \pipe\ROUTER \\.
Interfaces: 43

[...]

  8f09f000-b7ed-11ce-bbd2-00001a181cad v0.0

[...]
```

Table 4.95. rras operations

Interface	Operation number	Operation name
8f09f000-b7ed-11ce-bbd2-00001a181cad v0.0		
	0x00	RMprAdminServerGetInfo
	0x01	RRasAdminConnectionEnum
	0x02	RRasAdminConnectionGetInfo
	0x03	RRasAdminConnectionClearStats
	0x04	RRasAdminPortEnum
	0x05	RRasAdminPortGetInfo
	0x06	RRasAdminPortClearStats
	0x07	RRasAdminPortReset
	0x08	RRasAdminPortDisconnect
	0x09	RRouterInterfaceTransportSetGlobalInfo

	0x0a	RRouterInterfaceTransportGetGlobalInfo
	0x0b	RRouterInterfaceGetHandle
	0x0c	RRouterInterfaceCreate
	0x0d	RRouterInterfaceGetInfo
	0x0e	RRouterInterfaceSetInfo
	0x0f	RRouterInterfaceDelete
	0x10	RRouterInterfaceTransportRemove
	0x11	RRouterInterfaceTransportAdd
	0x12	RRouterInterfaceTransportGetInfo
	0x13	RRouterInterfaceTransportSetInfo
	0x14	RRouterInterfaceEnum
	0x15	RRouterInterfaceConnect
	0x16	RRouterInterfaceDisconnect
	0x17	RRouterInterfaceUpdateRoutes
	0x18	RRouterInterfaceQueryUpdateResult
	0x19	RRouterInterfaceUpdatePhonebookInfo
	0x1a	RMIBEntryCreate
	0x1b	RMIBEntryDelete
	0x1c	RMIBEntrySet
	0x1d	RMIBEntryGet
	0x1e	RMIBEntryGetFirst
	0x1f	RMIBEntryGetNext
	0x20	RMIBGetTrapInfo
	0x21	RMIBSetTrapInfo
	0x22	RRasAdminConnectionNotification
	0x23	RRasAdminSendUserMessage
	0x24	RRouterDeviceEnum
	0x25	RRouterInterfaceTransportCreate
	0x26	RRouterInterfaceDeviceGetInfo
	0x27	RRouterInterfaceDeviceSetInfo
	0x28	RRouterInterfaceSetCredentialsEx
	0x29	RRouterInterfaceGetCredentialsEx

	0x2a	RRasAdminConnectionRemoveQuarantine

4.11.20. Secondary Logon service

The Secondary Logon service runs one RPC service, available on the following endpoints:

- **SECLOGON** LPC port (Windows XP and later versions)
- **SecondaryLogon** named pipe (Windows 2000), **SECLOGON** named pipe (Windows XP)

```
Y:\>ifids -p ncalrpc -e SECLOGON serveur
Interfaces: 40

[...]

  12b81e99-f207-4a4c-85d3-77b42f76fd14 v1.0
```

Table 4.96. ISeclogon operations

Interface	Operation number	Operation name
12b81e99-f207-4a4c-85d3-77b42f76fd14 v1.0: ISeclogon		
	0x00	SeclCreateProcessWithLogonW
	0x01	SeclCreateProcessWithLogonExW

4.11.21. Security Configuration Editor Engine

The Security Configuration Editor Engine runs in the **services.exe** process context. It runs one RPC service on the following endpoint:

- **scerpc** named pipe

```
Y:\>ifids -p ncacn_np -e \pipe\scerpc \\.
Interfaces: 7

[...]

  93149ca2-973b-11d1-8c39-00c04fb984f9 v0.0

[...]
```

Table 4.97. SceSvc operations

Interface	Operation number	Operation name
93149ca2-973b-11d1-8c39-00c04fb984f9 v0.0: SceSvc		
	0x00	SceSvcRpcQueryInfo
	0x01	SceSvcRpcSetInfo
	0x02	SceRpcSetupUpdateObject
	0x03	SceRpcSetupMoveFile
	0x04	SceRpcGenerateTemplate
	0x05	SceRpcConfigureSystem
	0x06	SceRpcGetDatabaseInfo
	0x07	SceRpcGetObjectChildren
	0x08	SceRpcOpenDatabase
	0x09	SceRpcCloseDatabase
	0x0a	SceRpcGetDatabaseDescription

	0x0b	SceRpcGetDBTimeStamp
	0x0c	SceRpcGetObjectSecurity
	0x0d	SceRpcGetAnalysisSummary
	0x0e	SceRpcAnalyzeSystem
	0x0f	SceRpcUpdateDatabaseInfo
	0x10	SceRpcUpdateObjectInfo
	0x11	SceRpcStartTransaction
	0x12	SceRpcCommitTransaction
	0x13	SceRpcRollbackTransaction
	0x14	SceRpcGetServerProductType
	0x15	SceSvcRpcUpdateInfo
	0x16	SceRpcCopyObjects
	0x17	SceRpcSetupResetLocalPolicy
	0x18	SceRpcNotifySaveChangesInGP
	0x19	SceRpcControlNotificationQProcess
> Windows XP and Windows Server 2003	0x1a	SceRpcBrowseDatabaseTable
-	0x1b	SceRpcGetSystemSecurity
-	0x1c	SceRpcGetSystemSecurityFromHandle
-	0x1d	SceRpcSetSystemSecurity
-	0x1e	SceRpcSetSystemSecurityFromHandle
-	0x1f	SceRpcSetDatabaseSetting
-	0x20	SceRpcGetDatabaseSetting
-	0x21	SceRpcConfigureConvertedFileSecurityImmediately

4.11.22. SSDP Discovery Service service

Table 4.98. ssdpsrv operations

Interface	Operation number	Operation name
4b112204-0e19-11d3-b42b-0000f81feb9f v1.0: ssdpsrv		
	0x00	RegisterServiceRpc
	0x01	DeregisterServiceRpcByUSN
	0x02	DeregisterServiceRpc
	0x03	UpdateCacheRpc
	0x04	LookupCacheRpc
	0x05	CleanupCacheRpc
	0x06	InitializeSyncHandle
	0x07	RemoveSyncHandle
	0x08	RegisterNotificationRpc
	0x09	GetNotificationRpc
	0x0a	WakeupGetNotificationRpc
	0x0b	DeregisterNotificationRpc
	0x0c	EnableDeviceHost
	0x0d	DisableDeviceHost
	0x0e	SetICSInterfaces
	0x0f	SetICSOff

4.11.23. System Event Notification service

The System Event Notification Service runs two RPC service, listening on the following endpoint:

- **senssvc** LPC port

```
Y:\>ifids -p ncalrpc -e senssvc serveur
Interfaces: 43

[...]

  63fbe424-2029-11d1-8db8-00aa004abd5e v1.0
  629b9f66-556c-11d1-8dd2-00aa004abd5e v3.0

[...]
```

Table 4.99. SensApi operations

Interface	Operation number	Operation name
63fbe424-2029-11d1-8db8-00aa004abd5e v1.0: SensApi		
	0x00	Rpc_IsNetworkAlive
	0x01	Rpc_IsDestinationReachableW
	0x02	Rpc_IsDestinationReachableA

Table 4.100. SENSNotify operations

Interface	Operation number	Operation name
629b9f66-556c-11d1-8dd2-00aa004abd5e v3.0: SENSNotify		
	0x00	Rpc_SensNotifyWinlogonEvent
	0x01	Rpc_SensNotifyRasEvent

	0x02	Rpc_SensNotifyNetconEvent
	0x03	Rpc_SyncMgrExecCmd

4.11.24. Telephony service

The Telephony service runs one RPC service on the following endpoints:

- **tapsrvlpc** LPC port
- **tapsrv** named pipe

```
Y:\>ifids -p ncalrpc -e tapsrvlpc serveur
Interfaces: 1
  2f5f6520-ca46-1067-b319-00dd010662da v1.0

Y:\>ifids -p ncacn_np -e \pipe\tapsrv \\.
Interfaces: 1
  2f5f6520-ca46-1067-b319-00dd010662da v1.0
```

Table 4.101. tapsrv operations

Interface	Operation number	Operation name
2f5f6520-ca46-1067-b319-00dd010662da v1.0: tapsrv		
	0x00	ClientAttach
	0x01	ClientRequest
	0x02	ClientDetach

4.11.25. Terminal Server service

The Terminal Server service runs two RPC services, available on the following endpoints:

- **IcaApi** LPC port
- **LcRpc** LPC port
- **Ctx_WinStation_API_service** named pipe

```
Y:\>ifids -p ncalrpc -e IcaApi serveur
Interfaces: 2
  2f59a331-bf7d-48cb-9e5c-7c090d76e8b8 v1.0
  5ca4a760-ebb1-11cf-8611-00a0245420ed v1.0

Y:\>ifids -p ncalrpc -e LcRpc serveur
Interfaces: 2
  2f59a331-bf7d-48cb-9e5c-7c090d76e8b8 v1.0
  5ca4a760-ebb1-11cf-8611-00a0245420ed v1.0

Y:\>ifids -p ncacn_np -e \pipe\Ctx_Winstation_API_Service \\.
Interfaces: 2
  2f59a331-bf7d-48cb-9e5c-7c090d76e8b8 v1.0
  5ca4a760-ebb1-11cf-8611-00a0245420ed v1.0
```

The following interface is used for Terminal Services licensing:

Table 4.102. lcrpc operations

Interface	Operation number	Operation name
2f59a331-bf7d-48cb-9ec5-7c090d76e8b8 v1.0		
	0x00	RpcLicensingOpenServer
	0x01	RpcLicensingCloseServer
	0x02	RpcLicensingLoadPolicy
	0x03	RpcLicensingUnloadPolicy
	0x04	RpcLicensingSetPolicy

	0x05	RpcLicensingGetAvailablePolicyIds
	0x06	RpcLicensingGetPolicy
	0x07	RpcLicensingGetPolicyInformation
	0x08	RpcLicensingDeactivateCurrentPolicy

The following interface is used for Terminal Services remote management:

Table 4.103. winstation_rpc operations

Interface	Operation number	Operation name
5ca4a760-ebb1-11cf-8611-00a0245420ed v1.0		
	0x00	RpcWinStationOpenServer
	0x01	RpcWinStationCloseServer
	0x02	RpcIcaServerPing
	0x03	RpcWinStationEnumerate
	0x04	RpcWinStationRename
	0x05	RpcWinStationQueryInformation
	0x06	RpcWinStationSetInformation
	0x07	RpcWinStationSendMessage
	0x08	RpcLogonIdFromWinStationName
	0x09	RpcWinStationNameFromLogonId
	0x0a	RpcWinStationConnect
	0x0b	RpcWinStationVirtualOpen
	0x0c	RpcWinStationBeepOpen
	0x0d	RpcWinStationDisconnect
	0x0e	RpcWinStationReset
	0x0f	RpcWinStationShutdownSystem
	0x10	RpcWinStationWaitSystemEvent
	0x11	RpcWinStationShadow
	0x12	RpcWinStationShadowTargetSetup
	0x13	RpcWinStationShadowTarget
	0x14	RpcWinStationGenerateLicense

	0x15	RpcWinStationInstallLicense
	0x16	RpcWinStationEnumerateLicenses
	0x17	RpcWinStationActivateLicense
	0x18	RpcWinStationRemoveLicense
	0x19	RpcWinStationQueryLicense
	0x1a	RpcWinStationSetPoolCount
	0x1b	RpcWinStationQueryUpdateRequired
	0x1c	RpcWinStationCallback
	0x1d	RpcWinStationGetApplicationInfo
	0x1e	RpcWinStationReadRegistry
	0x1f	RpcWinStationWaitForConnect
	0x20	RpcWinStationNotifyLogon
	0x21	RpcWinStationNotifyLogoff
	0x22	RpcWinStationEnumerateProcesses
	0x23	RpcWinStationAnnoyancePopup
	0x24	RpcWinStationEnumerateProcesses
	0x25	RpcWinStationTerminateProcess
	0x26	RpcServerNWLogonSetAdmin
	0x27	RpcServerNWLogonQueryAdmin
	0x28	RpcWinStationNtsdDebug
	0x29	RpcWinStationBreakPoint
	0x2a	RpcWinStationCheckForApplicationName
	0x2b	RpcWinStationGetAllProcesses
	0x2c	RpcWinStationGetProcessSid
	0x2d	RpcWinStationGetTermSrvCountersValue
	0x2e	RpcWinStationReInitializeSecurity
	0x2f	RpcWinStationBroadcastSystemMessage
	0x30	RpcWinStationSendWindowMessage
	0x31	RpcWinStationNotifyNewSession
	0x32	RpcServerGetInternetConnectorStatus
	0x33	RpcServerSetInternetConnectorStatus
	0x34	RpcServerQueryInetConnectorInformation

	0x35	RpcWinStationGetLanAdapterName
> Windows XP and Windows Server 2003	0x36	RpcWinStationUpdateUserConfig
-	0x37	RpcWinStationQueryLogonCredentials
-	0x38	RpcWinStationRegisterConsoleNotification
-	0x39	RpcWinStationUnRegisterConsoleNotification
-	0x3a	RpcWinStationUpdateSettings
-	0x3b	RpcWinStationShadowStop
-	0x3c	RpcWinStationCloseServerEx
-	0x3d	RpcWinStationIsHelpAssistantSession
-	0x3e	RpcWinStationGetMachinePolicy
-	0x3f	RpcWinStationUpdateClientCachedCredentials
-	0x40	RpcWinStationFUSCanRemoteUserDisconnect
-	0x41	RpcWinStationCheckLoopBack
-	0x42	RpcConnectCallback
-	0x43	RpcWinStationNotifyDisconnectPipe
-	0x44	RpcWinStationSessionInitialized
-	0x45	RpcRemoteAssistancePrepareSystemRestore
-	0x46	RpcWinStationGetAllProcesses_NT6
-	0x47	RpcWinStationRegisterNotificationEvent
-	0x48	RpcWinStationUnRegisterNotificationEvent
-	0x49	RpcWinStationAutoReconnect
-	0x4a	RpcWinStationCheckAccess
-	0x4b	RpcWinStationOpenSessionDirectory

4.11.26. WebClient service

The WebClient service runs one RPC service, available on the following endpoint:

- **DAV RPC SERVICE** named pipe

```
Y:\>ifids -p ncacn_np -e "\pipe\DAV RPC SERVICE" \\.
Interfaces: 1
  c8cb7687-e6d3-11d2-a958-00c04f682e16 v1.0
```

Table 4.104. davclntrpc operations

Interface	Operation number	Operation name
c8cb7687-e6d3-11d2-a958-00c04f682e16 v1.0: davclntrpc		
	0x00	DavrCreateConnection
	0x01	DavrDoesServerDoDav
	0x02	DavrIsValidShare
	0x03	DavrEnumNetUses
	0x04	DavrEnumShares
	0x05	DavrEnumServers
	0x06	DavrGetConnection
	0x07	DavrDeleteConnection
	0x08	DavrGetUser
	0x09	DavrConnectionExist
	0x0a	DavrWinlogonLogonEvent
	0x0b	DavrWinlogonLogoffEvent
	0x0c	DavrGetDiskSpaceUsage
	0x0d	DavrFreeUsedDiskSpace
	0x0e	DavrGetTheLockOwnerOfTheFile

In February 2006, a security vulnerability discovered by Kostya Kortchinsky was fixed by Microsoft in the MS06-008 security bulletin [47]. It can be exploited (with valid user credentials) using the **DavrCreateConnection** operation (opnum 0) of the **davclntrpc** interface.

Prev	Up	Next
4.11.25. Terminal Server service	Home	4.11.27. Windows Audio service

4.11.27. Windows Audio service

The Windows Audio service runs one or several RPC services, avalailable on the following endpoints:

- **AudioSrv** LPC port
- **AudioSrv** named pipe (Windows XP before Service Pack 2)

In Windows Server 2003, the version of the **AudioSrv** interface is 1.0:

```
Y:\>ifids -p ncalrpc -e AudioSrv serveur
Interfaces: 40

[...]

  3faf4738-3a21-4307-b46c-fdda9bb8c0d5 v1.0

[...]
```

Table 4.105. AudioSrv operations

Interface	Operation number	Operation name
3faf4738-3a21-4307-b46c-fdda9bb8c0d5 v1.0: AudioSrv		
	0x00	gfxCreateZoneFactoriesList
	0x01	gfxCreateGfxFactoriesList
	0x02	gfxCreateGfxList
	0x03	gfxRemoveGfx
	0x04	gfxAddGfx
	0x05	gfxModifyGx
	0x06	gfxOpenGfx
	0x07	gfxLogon
	0x08	gfxLogoff

	0x09	winmmRegisterSessionNotificationEvent
	0x0a	winmmUnregisterSessionNotification
	0x0b	winmmSessionConnectState
	0x0c	wdmDriverOpenDrvRegKey
	0x0d	winmmAdvisePreferredDeviceChange
	0x0e	winmmGetPnpInfo

In Windows XP SP2 and >, the version of the **AudioSrv** interface is 1.1:

Table 4.106. AudioSrv operations

Interface	Operation number	Operation name
3faf4738-3a21-4307-b46c-fdda9bb8c0d5 v1.1: AudioSrv		
	0x00	gfxCreateZoneFactoriesList
	0x01	gfxCreateGfxFactoriesList
	0x02	gfxCreateGfxList
	0x03	gfxRemoveGfx
	0x04	gfxAddGfx
	0x05	gfxModifyGx
	0x06	gfxOpenGfx
	0x07	gfxLogon
	0x08	gfxLogoff
	0x09	wdmDriverOpenDrvRegKey
	0x0a	winmmAdvisePreferredDeviceChange
	0x0b	winmmGetPnpInfo

In Windows Vista, the **AudioSrv** interface (version 1.1) supports fewer operations. A new interface, **AudioRpc**, supports more operations:

```
Y:\>ifids -p ncalrpc -e AudioClientRpc vista
Interfaces: 9
  c386ca3e-9061-4a72-821e-498d83be188f v1.1
  3faf4738-3a21-4307-b46c-fdda9bb8c0d5 v1.1
```

[. . .]

Table 4.107. AudioRpc operations

Interface	Operation number	Operation name
c386ca3e-9061-4a72-821e-498d83be188f v1.1: AudioRpc		
	0x00	AudioServerConnect
	0x01	AudioServerDisconnect
	0x02	AudioServerInitialize
	0x03	AudioServerGetAudioSession
	0x04	AudioServerCreateStream
	0x05	AudioServerDestroyStream
	0x06	AudioServerGetStreamLatency
	0x07	AudioServerGetMixFormat
	0x08	AudioServerIsFormatSupported
	0x09	AudioServerGetDevicePeriod
	0x0a	AudioVolumeGetMasterVolumeLevelScalar
	0x0b	AudioSessionGetProcessId
	0x0c	AudioSessionGetState
	0x0d	AudioSessionGetLastActivation
	0x0e	AudioSessionGetLastInactivation
	0x0f	AudioSessionIsSystemSoundsSession
	0x10	AudioSessionGetDisplayName
	0x11	AudioSessionSetDisplayName
	0x12	AudioSessionGetSessionClass
	0x13	AudioSessionSetSessionClass
	0x14	AudioSessionGetVolume
	0x15	AudioSessionSetVolume
	0x16	AudioSessionGetMute

	0x17	AudioSessionSetMute
	0x18	AudioSessionGetChannelCount
	0x19	AudioSessionSetChannelVolume
	0x1a	AudioSessionGetChannelVolume
	0x1b	AudioSessionSetAllVolumes
	0x1c	AudioSessionGetAllVolumes
	0x1d	AudioServerDisconnect
	0x1e	AudioServerGetMixFormat
	0x1f	PolicyConfigGetDeviceFormat
	0x20	PolicyConfigSetDeviceFormat
	0x21	AudioServerGetDevicePeriod
	0x22	PolicyConfigSetProcessingPeriod
	0x23	PolicyConfigGetShareMode
	0x24	PolicyConfigSetShareMode
	0x25	GetAudioSessionManager
	0x26	AudioSessionManagerDestroy
	0x27	AudioSessionManagerGetAudioSession
	0x28	AudioSessionManagerGetCurrentSession
	0x29	AudioSessionManagerGetExistingSession
	0x2a	AudioSessionManagerAddAudioSessionClientNotification
	0x2b	AudioSessionManagerDeleteAudioSessionClientNotification
	0x2c	AudioSessionManagerAddAudioSessionClientNotification
	0x2d	AudioVolumeConnect
	0x2e	AudioVolumeDisconnect
	0x2f	AudioVolumeGetChannelCount
	0x30	AudioVolumeSetMasterVolumeLevel
	0x31	AudioVolumeSetMasterVolumeLevelScalar
	0x32	AudioVolumeGetMasterVolumeLevel
	0x33	AudioVolumeGetMasterVolumeLevelScalar
	0x34	AudioVolumeSetChannelVolumeLevel
	0x35	AudioVolumeSetChannelVolumeLevelScalar
	0x36	AudioVolumeGetChannelVolumeLevel

	0x37	AudioVolumeGetChannelVolumeLevelScalar
	0x38	AudioVolumeSetMute
	0x39	AudioSessionGetDisplayName
	0x3a	AudioVolumeAddMasterVolumeNotification
	0x3b	AudioVolumeDeleteMasterVolumeNotification
	0x3c	AudioMeterGetAverageRMS
	0x3d	AudioMeterGetChannelsRMS
	0x3e	AudioMeterGetPeakValue
	0x3f	AudioMeterGetChannelsPeakValues
	0x40	AudioVolumeGetStepInfo
	0x41	AudioVolumeStepUp
	0x42	AudioVolumeStepDown

Table 4.108. AudioSrv operations

Interface	Operation number	Operation name
3faf4738-3a21-4307-b46c-fdda9bb8c0d5 v1.1: AudioSrv		
	0x00	wdmDriverOpenDrvRegKey
	0x01	winmmAdvisePreferredDeviceChange
	0x02	winmmGetPnpInfo

4.11.28. Windows File Protection

The Windows File Protection subsystem runs inside the **winlogon.exe** process and supports one RPC service, available on the following endpoints:

- **SfcApi** LPC port
- **SfcApi** named pipe (Windows 2000 and Windows XP, not Windows Server 2003)

```
Y:\>ifids -p ncalrpc -e SfcApi serveur
Interfaces: 9

[...]

  83da7c00-e84f-11d2-9807-00c04f8ec850 v2.0

[...]
```

Table 4.109. sfcapi operations

Interface	Operation number	Operation name
83da7c00-e84f-11d2-9807-00c04f8ec850 v2.0: sfcapi		
	0x00	SfcSrv_GetNextProtectedFile
	0x01	SfcSrv_IsFileProtected
	0x02	SfcSrv_FileException
	0x03	SfcSrv_InitiateScan
	0x04	SfcSrv_PurgeCache
	0x05	SfcSrv_SetCacheSize
	0x06	SfcSrv_SetDisable
	0x07	SfcSrv_InstallProtectedFiles

4.11.29. Windows Security Center

The Security Center service was introduced in Windows XP SP2 and supports the **SecurityCenter** interface:

Table 4.110. SecurityCenter operations

Interface	Operation number	Operation name
06bba54a-be05-49f9-b0a0-30f790261023 v1.0: SecurityCenter		
	0x00	wscRegisterChangeNotification
	0x01	wscUnRegisterChangeNotification
	0x02	wscGetAlertStatus
	0x03	wscEnableComponentNotifications
	0x04	wscOverrideComponentStatus
	0x05	wscAutoUpdatesGetStatus
	0x06	wscAutoUpdatesEnableScheduledMode
	0x07	wscFirewallGetStatus
	0x08	wscIcfEnable
	0x09	wscAntivirusGetStatus
> Windows Vista	0x0a	wscAntiSpywareGetStatus
	0x0b	wscMsAsEnable
	0x0c	wscNotifyCurrentUserSettingChange
	0x0d	wscGeneralSecurityGetStatus
	0x0e	WscIeSettingsFix
	0x0f	WscLuaSettingsFix

4.11.30. Windows Time service

The Windows Time service runs one RPC service on the following endpoints:

- **W32TIME** LPC port (Windows 2000 and Windows XP) and **W32TIME_ALT** LPC port (Windows Server 2003, Windows Vista)
- **W32TIME** named pipe (Windows 2000 and Windows XP) and **W32TIME_ALT** named pipe (Windows Server 2003, Windows Vista)

```
Y:\>ifids -p ncalrpc -e W32TIME_ALT serveur
Interfaces: 40

[...]

  8fb6d884-2388-11d0-8c35-00c04fda2795 v4.1

[...]

Y:\>ifids -p ncacn_np -e \pipe\W32TIME_ALT \\.
Interfaces: 40

[...]

  8fb6d884-2388-11d0-8c35-00c04fda2795 v4.1

[...]
```

Table 4.111. w32time operations

Interface	Operation number	Operation name
8fb6d884-2388-11d0-8c35-00c04fda2795 v4.1: w32time		
-	0x00	W32TimeSync
-	0x01	W32TimeGetNetLogonServiceBits
> Windows XP and Windows Server 2003	0x02	W32TimeQueryProviderStatus

4.11.31. Winlogon process - Windows 2000

In Windows 2000, different RPC services run in the **winlogon.exe** process and are available on the following endpoints:

- **winlogonrpc** named pipe
- **InitShutdown** named pipe

```
C:\>ifids -p ncacn_np -e \pipe\winlogonrpc \\.
Interfaces: 4
  894de0c0-0d55-11d3-a322-00c04fa321a1 v1.0
  369ce4f0-0fdc-11d3-bde8-00c04f8eee78 v1.0
  a002b3a0-c9b7-11d1-ae88-0080c75e4ec1 v1.0
  83da7c00-e84f-11d2-9807-00c04f8ec850 v2.0

C:\>ifids -p ncacn_np -e \pipe\InitShutdown \\.
Interfaces: 4
  894de0c0-0d55-11d3-a322-00c04fa321a1 v1.0
  369ce4f0-0fdc-11d3-bde8-00c04f8eee78 v1.0
  a002b3a0-c9b7-11d1-ae88-0080c75e4ec1 v1.0
  83da7c00-e84f-11d2-9807-00c04f8ec850 v2.0
```

Table 4.112. InitShutdown operations

Interface	Operation number	Operation name
894de0c0-0d55-11d3-a322-00c04fa321a1 v1.0: InitShutdown		
	0x00	BaseInitiateShutdown
	0x01	BaseAbortShutdown
	0x02	BaseInitiateShutdownEx

profmap.dll DLL RPC service:

Table 4.113. pmapapi operations

Interface	Operation number	Operation name
369ce4f0-0fdc-11d3-bde8-00c04f8eee78 v1.0: pmapapi		
	0x00	ProfMapSrv_RemoteRemapUserProfile
	0x01	ProfMapSrv_RemoteRemapAndMoveUser

wlnotify.dll DLL RPC service:

Table 4.114. GetUserToken operations

Interface	Operation number	Operation name
a002b3a0-c9b7-11d1-ae88-0080c75e4ec1 v1.0: GetUserToken		
	0x00	SecpGetCurrentUserToken

The last interface is the sfcapi interface, documented in Section 4.11.28, “Windows File Protection”.

4.11.32. Winlogon process - Windows Server 2003

In Windows Server 2003, three additional RPC service are supported inside **winlogon.exe**. These interfaces create the following endpoints:

- **sclogonrpc** LPC port
- **IUserProfile** LPC port

```
Z:\>ifids -p ncalrpc -e sclogonrpc serveur
Interfaces: 9
  326731e3-c1c0-4a69-ae20-7d9044a4ea5c v1.0
  95958c94-a424-4055-b62b-b7f4d5c47770 v1.0
  894de0c0-0d55-11d3-a322-00c04fa321a1 v1.0
  83da7c00-e84f-11d2-9807-00c04f8ec850 v2.0
  a002b3a0-c9b7-11d1-ae88-0080c75e4ec1 v1.0
  00000134-0000-0000-c000-000000000046 v0.0
  18f70770-8e64-11cf-9af1-0020af6e72f4 v0.0
  00000131-0000-0000-c000-000000000046 v0.0
  00000143-0000-0000-c000-000000000046 v0.0

Z:\>ifids -p ncalrpc -e IUserProfile serveur
Interfaces: 9
  326731e3-c1c0-4a69-ae20-7d9044a4ea5c v1.0
  95958c94-a424-4055-b62b-b7f4d5c47770 v1.0
  894de0c0-0d55-11d3-a322-00c04fa321a1 v1.0
  83da7c00-e84f-11d2-9807-00c04f8ec850 v2.0
  a002b3a0-c9b7-11d1-ae88-0080c75e4ec1 v1.0
  00000134-0000-0000-c000-000000000046 v0.0
  18f70770-8e64-11cf-9af1-0020af6e72f4 v0.0
  00000131-0000-0000-c000-000000000046 v0.0
  00000143-0000-0000-c000-000000000046 v0.0
```

userenv.dll DLL RPC service:

Table 4.115. IUserProfile operations

Interface	Operation number	Operation name
326731e3-c1c0-4a69-ae20-7d9044a4ea5c v1.0: IUserProfile		
	0x00	DropClientContext
	0x01	LoadUserProfileI
	0x02	UnloadUserProfileI
	0x03	ReleaseClientContext
	0x04	EnterUserProfileLockRemote
	0x05	LeaveUserProfileLockRemote

Table 4.116. IProfileDialog operations

Interface	Operation number	Operation name
4825ea41-51e3-4c2a-8406-8f2d2698395f v1.0: IProfileDialog		
	0x00	ErrorDialog
	0x01	SlowLinkDialog

Table 4.117. IRPCSCLogon operations

Interface	Operation number	Operation name
95958c94-a424-4055-b62b-b7f4d5c47770 v1.0: IRPCSCLogon		
	0x00	RPC_ScHelperInitializeContext
	0x01	RPC_ScHelperRelease
	0x02	RPC_ScHelperGetCertFromLogonInfo
	0x03	RPC_ScHelperGetProvParam
	0x04	RPC_ScHelperGenRandBits
	0x05	RPC_ScHelperVerifyCardAndCreds
	0x06	RPC_ScHelperEncryptCredentials
	0x07	RPC_ScHelperSignPkcsMessage
	0x08	RPC_ScHelperDecryptMessage
	0x09	RPC_ScHelper_CryptAcquireCertificatePrivateKey

	0x0a	RPC_ScHelper_CryptSetProvParam
	0x0b	RPC_ScHelper_CryptReleaseContext

Prev | Up | Next

4.11.31. Winlogon process - Windows 2000 | Home | 4.11.33. Wireless Configuration service

4.11.33. Wireless Configuration service

The Wireless Configuration service runs one RPC service, available on the following endpoint:

- **wzcsvc** LPC port

```
X:\>ifids -p ncalrpc -e wzcsvc serveur
Interfaces: 37
  621dff68-3c39-4c6c-aae3-e68e2c6503ad v1.0

[...]
```

Table 4.118. winwzc operations

Interface	Operation number	Operation name
621dff68-3c39-4c6c-aae3-e68e2c6503ad v1.0: winwzc		
	0x00	RpcGetAPIVersion
	0x01	RpcEnumInterfaces
	0x02	RpcQueryInterface
	0x03	RpcSetInterface
	0x04	RpcRefreshInterface
	0x05	RpcQueryContext
	0x06	RpcSetContext
	0x07	RpcEapolUIResponse
	0x08	RpcEapolGetCustomAuthData
	0x09	RpcEapolSetCustomAuthData
	0x0a	RpcEapolGetInterfaceParams
	0x0b	RpcEapolSetInterfaceParams
	0x0c	RpcEapolGetEapUserInfo
	0x0d	RpcEapolSetUserInfo

	0x0e	RpcEapolReAuthenticateInterface
	0x0f	RpcEapolQueryInterfaceState
	0x10	RpcOpenWZCDbLogSession
	0x11	RpcCloseWZCDbLogSession
	0x12	RpcEnumWZCDbLogRecords
	0x13	RpcFlushWZCDbLog
	0x14	RpcGetWZCDbLogRecord

Prev | Up | Next

4.11.32. Winlogon process - Windows Server 2003 | Home | 4.12. MSRPC interfaces introduced in Windows Vista

4.12. MSRPC interfaces introduced in Windows Vista

4.12.1. Group Policy Client Service
4.12.2. Network Location Awareness
4.12.3. Network Store Interface
4.12.4. Parental controls
4.12.5. Peer Networking Identity Manager
4.12.6. Remote Registry Service
4.12.7. Windows event collector service
4.12.8. Windows event logging service
4.12.9. Windows Firewall
4.12.10. Windows Wireless LAN 802.11 Auto Configuration Service
4.12.11. Wired Autoconfiguration Service

4.12.1. Group Policy Client Service

Table 4.119. IGroupPolicyUtilities operations

Interface	Operation number	Operation name
2eb08e3e-639f-4fba-97b1-14f878961076 v1.0: IGroupPolicyUtilities		
	0x00	ProcessRefresh
	0x01	RegisterForNotification
	0x02	CheckRegisterForNotification
	0x03	LockPolicySection
	0x04	UnLockPolicySection

4.12.2. Network Location Awareness

```
Y:\>ifids -p ncalrpc -e nlaapi vista
Interfaces: 11

[...]

  aa411582-9bdf-48fb-b42b-faa1eee33949 v1.0
  c33b9f46-2088-4dbc-97e3-6125f127661c v1.0

[...]

Y:\>ifids -p ncalrpc -e nlaplg vista
Interfaces: 11

[...]

  aa411582-9bdf-48fb-b42b-faa1eee33949 v1.0
  c33b9f46-2088-4dbc-97e3-6125f127661c v1.0

[...]
```

Table 4.120. nlaapi operations

Interface	Operation number	Operation name
c33b9f46-2088-4dbc-97e3-6125f127661c v1.0: nlaapi		
	0x00	NlaOpenQuery
	0x01	NlaAsyncIndicate
	0x02	NlaRefreshQuery
	0x03	NlaCloseQuery

Table 4.121. nlaplg operations

Interface	Operation number	Operation name

aa411582-9bdf-48fb-b42b-faa1eee33949 v1.0: nlaplg		
	0x00	RpcPluginRegister
	0x01	RpcPluginUnregister
	0x02	RpcPluginDataIndicate

4.12.3. Network Store Interface

Table 4.122. WinNsi operations

Interface	Operation number	Operation name
7ea70bcf-48af-4f6a-8968-6a440754d5fa v1.0: WinNsi		
	0x00	NsiGetParameter
	0x01	NsiGetAllParameters
	0x02	NsiEnumerateObjectsAllParameters
	0x03	NsiSetParameter
	0x04	NsiSetAllParameters
	0x05	NsiRegisterChangeNotification
	0x06	NsiDeregisterChangeNotification
	0x07	NsiRequestChangeNotification
	0x08	NsiParameterChange

4.12.4. Parental controls

Table 4.123. WPCSvc operations

Interface	Operation number	Operation name
1dfce5a8-dd8a-aace-f603922fd9e7 v0.1: WPCSvc		
	0x00	WPCIsCurrentProcessHTTPFiltered
	0x01	WPCIsURLBlocked
	0x02	WPCIsHTTPRequestBlocked

4.12.5. Peer Networking Identity Manager

Table 4.124. IP2pIMSvc operations

Interface	Operation number	Operation name
8174bb16-571b-4c38-8386-1102b449044a v1.0: IP2pIMSvc		
	0x00	PeerIdentityCreate
	0x01	PeerIdentityCreateDefault
	0x02	PeerIdentityRelease
	0x03	PeerIdentityDelete
	0x04	PeerIdentityByPeerName
	0x05	PeerIdentitySetFriendlyName
	0x06	PeerIdentityCreateCert
	0x07	PeerIdentityGetCert
	0x08	PeerIdentitySetCert
	0x09	PeerIdentityList
	0x0a	PeerIdentityGroupCreate
	0x0b	PeerIdentityGroupList
	0x0c	PeerIdentityGroupDelete
	0x0d	PeerIdentityImport

Table 4.125. IPeerGroupSvc operations

Interface	Operation number	Operation name
3f31c91e-2545-4b7b-9311-9529e8bffef6 v1.0: IPeerGroupSvc		
	0x00	PeerGroupCreateRpc
	0x01	PeerGroupOpenRpc
	0x02	PeerGroupJoinRpc

	0x03	PeerGroupPasswordJoinRpc
	0x04	PeerGroupConnectRpc
	0x05	PeerGroupCloseRpc
	0x06	PeerGroupDeleteRpc
	0x07	PeerGroupCreateInvitationRpc
	0x08	PeerGroupCreatePasswordInvitationRpc
	0x09	PeerGroupGetStatusRpc
	0x0a	PeerGroupGetPropertiesRpc
	0x0b	PeerGroupSetPropertiesRpc
	0x0c	PeerGroupEnumConnectionsRpc
	0x0d	PeerGroupEnumMembersRpc
	0x0e	PeerGroupOpenDirectConnectionRpc
	0x0f	PeerGroupCloseDirectConnectionRpc
	0x10	PeerGroupSendDataRpc
	0x11	PeerGroupRegisterEventRpc
	0x12	PeerGroupUnregisterEventRpc
	0x13	PeerGroupGetEventDataRpc
	0x14	PeerGroupGetRecordRpc
	0x15	PeerGroupAddRecordRpc
	0x16	PeerGroupUpdateRecordRpc
	0x17	PeerGroupDeleteRecordRpc
	0x18	PeerGroupEnumRecordsRpc
	0x19	PeerGroupSearchRecordsRpc
	0x1a	PeerGroupExportDatabaseRpc
	0x1b	PeerGroupImportDatabaseRpc
	0x1c	PeerGroupUniversalTimeToPeerTimeRpc
	0x1d	PeerGroupTimeToUniversalTimeRpc
	0x1e	PeerGetItemCountRpc
	0x1f	PeerGetNextMemberInfoItemRpc
	0x20	PeerGetNextConnectionInfoItemRpc
	0x21	PeerGetNextRecordItemRpc
	0x22	PeerEndEnumerationRpc

	0x23	PeerGroupIssueCredentialsRpc	

Table 4.126. pnrpsvc operations

Interface	Operation number	Operation name
a2d47257-12f7-4beb-8981-0ebfa935c407 v1.0: pnrpsvc		
	0x00	InitializeCloudList
	0x01	GetCloudList
	0x02	ShutdownCloudList
	0x03	ClientInitialize
	0x04	ClientUninitialize
	0x05	Register
	0x06	Unregister
	0x07	Resolve
	0x08	GetResolveResult
	0x09	EndResolve
	0x0a	Ping
	0x0b	DiagControl

4.12.6. Remote Registry Service

Table 4.127. perflibv2 operations

Interface	Operation number	Operation name	Windows API
0da5a86c5-12c2-4943-30ab-7f74a813d853 v1.0: perflibv2			
	0x00	PerflibV2EnumerateCounterSet	
	0x01	PerflibV2QueryCounterSetRegistrationInfo	
	0x02	PerflibV2EnumerateCounterSetInstance	
	0x03	PerflibV2OpenQueryHandle	
	0x04	PerflibV2CloseQueryHandle	
	0x05	PerflibV2QueryCounterInfo	
	0x06	PerflibV2QueryCounterData	
	0x07	PerflibV2ValidateCounters	

4.12.7. Windows event collector service

Table 4.128. ICollectorService operations

Interface	Operation number	Operation name	Windows API
69510fa1-2f99-4eeb-a4ff-af259f0f9749 v1.0: ICollectorService			
	0x00	EcRpcGetSubscriptionNames	
	0x01	EcRpcGetSubscription	
	0x02	EcRpcPutSubscription	
	0x03	EcRpcDeleteSubscription	
	0x04	EcRpcGetSubscriptionRunTimeStatus	
	0x05	EcRpcRetrySubscription	

4.12.8. Windows event logging service

The **IEventService** interface is the RPC interface used to communicate with the **Windows event logging service** service.

The interface is used over a dynamic TCP endpoint, registered in the endpoint mapper database, as shown below:

```
[...]

IfId: f6beaff7-1e19-4fbb-9f8f-b89e2018337c version 1.0
Annotation: Event log TCPIP
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:127.0.0.1[\\pipe\\eventlog]

IfId: f6beaff7-1e19-4fbb-9f8f-b89e2018337c version 1.0
Annotation: Event log TCPIP
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:127.0.0.1[49153]

[...]
```

For more information about the Windows Event Log API, see the documentation in the Microsoft Windows SDK.

Table 4.129. IEventService operations

Interface	Operation number	Operation name	Windows API
f6beaff7-1e19-4fbb-9f8f-b89e2018337c v1.0: IEventService			
	0x00	EvtRpcRegisterRemoteSubscription	
	0x01	EvtRpcUpdateRemoteSubscription	

	0x02	EvtRpcRemoteSubscriptionNextAsync	
	0x03	EvtRpcRemoteSubscriptionNext	
	0x04	EvtRpcRemoteSubscriptionWaitAsync	
	0x05	EvtRpcRegisterLogQuery	
	0x06	EvtRpcClearLog	
	0x07	EvtRpcExportLog	
	0x08	EvtRpcLocalizeExportLog	
	0x09	EvtRpcMessageRender	
	0x0a	EvtRpcMessageRenderDefault	
	0x0b	EvtRpcQueryNext	
	0x0c	EvtRpcQuerySeek	
	0x0d	EvtRpcClose	
	0x0e	EvtRpcAssertConfig	
	0x0f	EvtRpcRetractConfig	
	0x10	EvtRpcOpenLogHandle	
	0x11	EvtRpcGetLogFileInfo	
	0x12	EvtRpcGetChannelList	
	0x13	EvtRpcGetChannelConfig	
	0x14	EvtRpcPutChannelConfig	
	0x15	EvtRpcGetPublisherList	
	0x16	EvtRpcGetPublisherMetadata	
	0x17	EvtRpcGetPublisherResourceMetadata	
	0x18	EvtRpcGetEventMetadataEnum	
	0x19	EvtRpcGetNextEventMetadata	

4.12.9. Windows Firewall

Table 4.130. FwRpc operations

Interface	Operation number	Operation name
2fb92682-6599-42dc-ae13-bd2ca89bd11c v1.0: FwRpc		
	0x00	FWOpenPolicyStore
	0x01	FWClosePolicyStore
	0x02	FWRestoreDefaults
	0x03	FWGetGlobalConfig
	0x04	FWSetGlobalConfig
	0x05	FWAddFirewallRule
	0x06	FWSetFirewallRule
	0x07	FWDeleteFirewallRule
	0x08	FWDeleteAllFirewallsRules
	0x09	FWEnumFirewallRules
	0x0a	FWGetConfig
	0x0b	FWSetConfig
	0x0c	FWAddConnectionSecurityRule
	0x0d	FWSetConnectionSecurityRule
	0x0e	FWDeleteConnectionSecurityRule
	0x0f	FWDeleteAllConnectionSecurityRules
	0x10	FWEnumConnectionSecurityRules
	0x11	FWAddAuthenticationSet
	0x12	FWSetAuthenticationSet
	0x13	FWDeleteAuthenticationSet
	0x14	FWDeleteAllAuthenticationSets
	0x15	FWEnumAuthenticationSets

	0x16	FWAddCryptoSet
	0x17	FWSetCryptoSet
	0x18	FWDeleteCryptoSet
	0x19	FWDeleteAllCryptoSets
	0x1a	FWEnumCryptoSets
	0x1b	FWEnumPhase1SAs
	0x1c	FWEnumPhase2SAs
	0x1d	FWDeletePhase1SAs
	0x1e	FWDeletePhase2SAs

Table 4.131. Fw_Resource_Indication operations

Interface	Operation number	Operation name
7f9d11bf-7fb9-436b-a812-b2d50c5d4c03 v1.0: Fw_Resource_Indication		
	0x00	FWIndicatePortInUse
	0x01	FWGetIndicatedPortInUse

4.12.10. Windows Wireless LAN 802.11 Auto Configuration Service

Table 4.132. winwlan operations

Interface	Operation number	Operation name	Windows API
266f33b4-c7c1-4bd1-8f52-ddb8f2214ea9 v1.0: winwlan			
	0x00	RpcOpenHandle	WlanOpenHandle
	0x01	RpcCloseHandle	WlanCloseHandle
	0x02	RpcEnumInterfaces	WlanEnumInterfaces
	0x03	RpcSetAutoConfigParameter	WlanSetAutoConfigParameter
	0x04	RpcQueryAutoConfigParameter	WlanQueryAutoConfigParameter
	0x05	RpcGetInterfaceCapability	WlanGetInterfaceCapability
	0x06	RpcSetInterface	WlanSetInterface
	0x07	RpcQueryInterface	WlanQueryInterface
	0x08	RpcScan	WlanScan
	0x09	RpcGetVisibleNetworkList	WlanGetVisibleNetworkList
	0x0a	RpcGetNetworkBssList	WlanGetNetworkBssList
	0x0b	RpcConnect	WlanConnect
	0x0c	RpcDisconnect	WlanDisconnect
	0x0d	RpcRegisterNotification	WlanRegisterNotification
	0x0e	RpcAsynGetNotification	
	0x0f	RpcSetProfile	WlanSetProfile
	0x10	RpcGetProfile	WlanGetProfile
	0x11	RpcDeleteProfile	WlanDeleteProfile

	0x12	RpcSetProfileList	WlanSetProfileList
	0x13	RpcGetProfileList	WlanGetProfileList
	0x14	RpcSetProfilePosition	WlanSetProfilePosition
	0x15	RpcSetUserData	WlanSetUserData
	0x16	RpcSetFilterList	WlanSetFilterList
	0x17	RpcGetFilterList	WlanGetFilterList
	0x18	RpcSetPsdlEDataList	WlanSetPsdIEDataList
	0x19	RpcSaveTemporaryProfile	WlanSaveTemporaryProfile
	0x1a	RpcIsUIRequestPending	
	0x1b	RpcSetUIForwardingNetworkList	
	0x1c	RpcIsNetworkSuppressed	
	0x1d	RpcRemoveUIForwardingNetworkList	
	0x1e	RpcQueryExtUIRequest	
	0x1f	RpcUIReponse	
	0x20	RpcGetProfileKeyInfo	

Table 4.133. winwdiag operations

Interface	Operation number	Operation name
25952c5d-7976-4aa1-a3cb-c35f7ae79d1b v1.0: winwdiag		
	0x00	WDiagRpcInitSession
	0x01	WDiagRpcEndSession
	0x02	WDiagRpcQueryStatistics
	0x03	WDiagRpcQueryConnectionContextInfo
	0x04	WDiagRpcAutoConfigGetInterfaceInfo
	0x05	WDiagRpcGetExtensibilityInfo

4.12.11. Wired Autoconfiguration Service

Table 4.134. winlan operations

Interface	Operation number	Operation name
1bddb2a6-c0c3-41be-8703-ddbdf4f0e80a v1.0: winlan		
	0x00	RpcOpenHandle
	0x01	RpcCloseHandle
	0x02	RpcEnumInterfaces
	0x03	RpcSetInterface
	0x04	RpcGetInterfaceState
	0x05	RpcReConnect
	0x06	RpcRegisterNotification
	0x07	RpcAsynGetNotification
	0x08	RpcSetProfile
	0x09	RpcGetProfile
	0x0a	RpcGetCurrentProfile
	0x0b	RpcDeleteProfile
	0x0c	RpcUIReponse
	0x0d	RpcQueryUIRequest

4.13. Implication of multiple RPC services in one process

4.13.1. Win32 services hosting
4.13.2. Example of multiple RPC services in one process
4.13.3. Implications of running multiple RPC services in one process

One important property of the MSRPC implementation is that inside a given process any RPC services listening on any protocol sequences can be reached using any opened endpoints.

This specifity of the MSRPC implementation is documented in the MSDN, under the “Be Wary of Other RPC Endpoints Running in the Same Process” [90] section.

To restrict RPC calls on the server-side, either the **RpcServerRegister2()** API or the **RpcServerRegisterIfEx()** API must be used.

As most Win32 services are implemented in a few processes, hosting many Win32 services (**lsass.exe**, **services.exe**, **svchost.exe**), a direct consequence is that all RPC services started by any Win32 service in a given process can be invoked using any opened endpoint in the process context, if RPC services do not use one of the two APIs earlier mentionned.

4.13.1. Win32 services hosting

The services.exe process host many services, which can be identified looking for **services.exe** in the following registry value of each service **service_name**:

```
Key: HKLM\SYSTEM\CurrentControlSet\Services\service_name\
Value: ImagePath
```

Three instances of svchost.exe processes can be found on a Windows 2000 system. Among them, one instance (netsvcs instance) typically hosts different services. Services hosted in svchost.exe processes appear in the registry:

```
Key: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost\
Values: netsvcs, rpcss, tapisrv
```

More precisely, on Windows 2000 systems, the following Win32 services run in the following processes:

- lsass.exe: kdc, netlogon, NtLmSsp, PolicyAgent, SamSs
- services.exe: Alerter, AppMgmt, Browser, Dhcp, dmserver, Dnscache, Eventlog, lanmanserver, lanmanworkstation, LmHosts, Messenger, PlugPlay, ProtectedStorage, seclogon, TrkSvr, TrkWks, W32Time, Wmi
- svchost.exe (netsvcs instance): EventSystem, Ias, Iprip, Irmon, Netman, Nwsapagent, Rasauto, Rasman, Remoteaccess, SENS, Sharedaccess, Ntmssvc
- svchost.exe (rpcss instance): rpcss
- svchost.exe (tapisrv instance): Tapisrv

On Windows XP systems, Win32 services run in the following processes:

- lsass.exe: Netlogon, NtLmSsp, PolicyAgent, ProtectedStorage, SamSs
- services.exe: Eventlog, PlugPlay
- svchost.exe (LocalService instance, running as LocalService): Alerter, WebClient, LmHosts, RemoteRegistry, upnphost, SSDPSRV
- svchost.exe (NetworkService instance, running as NetworkService): DnsCache
- svchost.exe (netsvcs instance): 6to4, AppMgmt, AudioSrv, Browser, CryptSvc, DMServer, DHCP, ERSvc, EventSystem, FastUserSwitchingCompatibility, HidServ, Ias, Iprip, Irmon, LanmanServer, LanmanWorkstation, Messenger, Netman, Nla, Ntmssvc, NWCWorkstation,

Nwsapagent, Rasauto, Rasman, Remoteaccess, Schedule, Seclogon, SENS, Sharedaccess, SRService, Tapisrv, Themes, TrkWks, W32Time, WZCSVC, Wmi, WmdmPmSp, winmgmt, TermService, wuauserv, BITS, ShellHWDetection, helpsvc, uploadmgr
- svchost.exe (rpcss instance): rpcss
- svchost.exe (termsvcs instance): TermService
- svchost.exe (imgsvc instance); StiSvc

On Windows Server 2003 systems, Win32 services are organized as follow:

- lsass.exe: HTTPFilter, kdc, Netlogon, NtLmSsp, PolicyAgent, ProtectedStorage, SamSs
- services.exe: Eventlog, PlugPlay
- svchost.exe (LocalService instance, running as LocalService): Alerter, WebClient, LmHosts, WinHttpAutoProxySvc
- svchost.exe (NetworkService instance, running as NetworkService): 6to4, DHCP, DnsCache
- svchost.exe (netsvcs instance): AppMgmt, AudioSrv, Browser, CryptSvc, DMServer, EventSystem, HidServ, Ias, Iprip, Irmon, LanmanServer, LanmanWorkstation, Messenger, Netman, Nla, Ntmssvc, NWCWorkstation, Nwsapagent, Rasauto, Rasman, Remoteaccess, Sacsvr, Schedule, Seclogon, SENS, Sharedaccess, Themes, TrkWks, TrkSvr, W32Time, WZCSVC, Wmi, WmdmPmSp, winmgmt, wuauserv, BITS, ShellHWDetection, helpsvc, uploadmgr, WmdmPmSN
- svchost.exe (rpcss instance): rpcss
- svchost.exe (regsvc instance): RemoteRegistry
- svchost.exe (swprv instance): swprv
- svchost.exe (tapisrv instance): Tapisrv
- svchost.exe (termsrv instance): TermService
- svchost.exe (WinErr instance): ERsvc
- svchost.exe (imgsvc instance): StiSvc

To determine which services are hosted by which services on a running system, the following tools can be used:

- the Process Explorer tool [91]
- option **/s** of the tlist utility (part of Windows 2000 support tools)
- option **/svc** of the tasklist utility (available in Windows XP and later)

4.13.2. Example of multiple RPC services in one process

Using ifids with the **eventlog** named pipe endpoint, opened by the Eventlog service running in the **services.exe** process, the list of interface identifiers is:

```
C:\WINNT>ifids -p ncacn_np -e \pipe\eventlog \\.
Interfaces: 13
  367abb81-9844-35f1-ad32-98f038001003 v2.0
  93149ca2-973b-11d1-8c39-00c04fb984f9 v0.0
  82273fdc-e32a-18c3-3f78-827929dc23ea v0.0
  65a93890-fab9-43a3-b2a5-1e330ac28f11 v2.0
  8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0
  8d0ffe72-d252-11d0-bf8f-00c04fd9126b v1.0
  c9378ff1-16f7-11d0-a0b2-00aa0061426a v1.0
  0d72a7d4-6148-11d1-b4aa-00c04fb66ea0 v1.0
  4b324fc8-1670-01d3-1278-5a47bf6ee188 v3.0
  6bffd098-a112-3610-9833-46c3f87e345a v1.0
  17fdd703-1827-4e34-79d4-24a55c53bb37 v1.0
  5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc v1.0
  8fb6d884-2388-11d0-8c35-00c04fda2795 v4.1
```

Using another endpoint, for example, the dynamic UDP port opened by the messenger service (also running in the **services.exe** process), the result is identical:

```
C:\WINNT>ifids -p ncadg_ip_udp -e 1026 127.0.0.1
Interfaces: 13
  367abb81-9844-35f1-ad32-98f038001003 v2.0
  93149ca2-973b-11d1-8c39-00c04fb984f9 v0.0
  82273fdc-e32a-18c3-3f78-827929dc23ea v0.0
  65a93890-fab9-43a3-b2a5-1e330ac28f11 v2.0
  8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0
  8d0ffe72-d252-11d0-bf8f-00c04fd9126b v1.0
  c9378ff1-16f7-11d0-a0b2-00aa0061426a v1.0
  0d72a7d4-6148-11d1-b4aa-00c04fb66ea0 v1.0
  4b324fc8-1670-01d3-1278-5a47bf6ee188 v3.0
  6bffd098-a112-3610-9833-46c3f87e345a v1.0
  17fdd703-1827-4e34-79d4-24a55c53bb37 v1.0
  5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc v1.0
  8fb6d884-2388-11d0-8c35-00c04fda2795 v4.1
```

These results show that all RPC services in the **services.exe** process can be reached using any opened endpoint on any transport.

Using our knowledge of RPC interface identifiers, we can identify some of the Win32 services currently running in the **services.exe** process:

```
C:\WINNT>ifids -p ncadg_ip_udp -e 1026 127.0.0.1
```

```
Interfaces: 13
  367abb81-9844-35f1-ad32-98f038001003 v2.0  Services Control Manager (SCM)
  93149ca2-973b-11d1-8c39-00c04fb984f9 v0.0  Security Configuration Editor (SCE)
  82273fdc-e32a-18c3-3f78-827929dc23ea v0.0  Eventlog service
  65a93890-fab9-43a3-b2a5-1e330ac28f11 v2.0  DNS Client service (Windows 2000)
  8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0  Plug and Play service
  8d0ffe72-d252-11d0-bf8f-00c04fd9126b v1.0  |
  c9378ff1-16f7-11d0-a0b2-00aa0061426a v1.0  |__ Protected Storage service
  0d72a7d4-6148-11d1-b4aa-00c04fb66ea0 v1.0  |
  4b324fc8-1670-01d3-1278-5a47bf6ee188 v3.0  Server service
  6bffd098-a112-3610-9833-46c3f87e345a v1.0  Workstation service
  17fdd703-1827-4e34-79d4-24a55c53bb37 v1.0  |__ Messenger service
  5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc v1.0  |
  8fb6d884-2388-11d0-8c35-00c04fda2795 v4.1  Windows Time service
```

Thus, the following Win32 services are running:

- Eventlog
- Dnscache
- ProtectedStorage
- lanmanserver
- lanmanworkstation
- Messenger
- PlugPlay
- W32Time

Actually, the complete list of Win32 services running inside the **services.exe** process is:

```
C:\WINNT>tlist /s

[...]

 256 SERVICES.EXE    Svcs:  Alerter,Dnscache,Eventlog,lanmanserver,lanmanworkstation,
LmHosts,Messenger,PlugPlay,ProtectedStorage,seclogon,W32Time

[...]
```

4.13.3. Implications of running multiple RPC services in one process

The direct consequence of running multiple RPC services in one process is that, if one RPC service is listening on an endpoint like a TCP port or a named pipe, all RPC services can be reached using that particular endpoint.

Thus, even if a RPC service listens only on the ncalrpc protocol (in order to accept only local procedure calls), it can be used remotely as long as another RPC service in the same process listens on a TCP port or a named pipe.

Another consequence is that it allows to anonymously identify some Win32 services remotely, as shown in the previous section:

- Services running in the **lsass.exe** process can be identified, using the **lsarpc**, **samr**, **netlogon** named pipes as MSRPC endpoint (these named pipes can always be opened in the context of a SMB NULL session)
- Some services in the **services.exe** process can be identified, using either the dynamic UDP port opened by the messenger service as MSRPC endpoint or the **wkssvc**, **srvsvc** or **browser** named pipes (they can always be opened in the context of a SMB NULL session).
- Identifying Win32 services running in **svchost.exe** instances can be more difficult, in particular when RPC services contained in that processes do not open endpoints that can be used remotely.

Note: this also explains why one RPC interface identifier can appear more than once in the rpcdump output, with different endpoints on different protocol sequences: these correspond to endpoints opened by RPC services in the same process.

4.14. RPC services protection

Developpers of RPC services can protect their applications against the problem described in the previous section using two new API, **RpcServerRegisterIfEx()** and **RpcServerRegisterIf2()**. These new API allow the specification of a security-callback function, on a per-interface basis.

Typically, a security-callback function verifies that the protocol sequence used by a client is legal. For instance, it is thus possible to forbid access to RPC services that are supposed to be used only locally, even if the process that hosts RPC services also runs RPC services listening on named pipes or TCP or UDP sockets.

When these APIs are used, it usually implies that only a subset of all the interfaces that appear on the output of the **ifids** command can be reached, using dedicated endpoints.

4.15. RPC interfaces restriction in Windows XP SP2, Windows Server 2003 SP1 and later versions

In Windows XP SP2, RPC interfaces restrictions were introduced. By default, it is not possible to bind anonymously to an interface, except using the **ncacn_np** protocol sequence.

The **RestrictRemoteClients** registry value (Restrictions for Unauthenticated RPC Client GPO), not present by default but with a default value of 1 (**RPC_RESTRICT_REMOTE_CLIENT_DEFAULT**) prevents anonymous binding to RPC interfaces, typically using the **ncacn_ip_tcp** transport.

In particular, the restriction applies to RPC interfaces running in the rpcss service, including the endpoint mapper (**epmp**) interface. Yet, it is still possible to query the endpoint mapper anonymously using the **ncacn_np** protocol sequence with the **\pipe\epmapper** named pipe as endpoint.

For more information, see the RPC Interface Restriction section in the Network Protection technologies chapter in the Changes to Functionality in Microsoft Windows XP Service Pack 2 document [94] and the #838191 Microsoft knowledge base article [95].

The **RestrictRemoteClients** registry value (Restrictions for Unauthenticated RPC Client GPO) also exists in Windows Server 2003 SP1 but its default value is 0 (restrictions are disabled). It is recommended to set it to 1 (**RPC_RESTRICT_REMOTE_CLIENT_DEFAULT**) or 2 (**RPC_RESTRICT_REMOTE_CLIENT_HIGH**).

4.16. MSRPC vulnerabilities

Over the last years, several security vulnerabilities were discovered in the Windows MSRPC subsystem.

Follows a list of Microsoft security patches that fixed vulnerabilities related to MSRPC, either in the MSRPC subsystem or in system components running MSRPC services:

- MS98-014: RPC Spoofing Denial of Service on Windows NT (CVE-1999-0969)
- MS98-017: Named Pipes Over RPC Vulnerability (CVE-1999-1127)
- MS99-020: Malformed LSA Request Vulnerability (CVE-1999-0721)
- MS99-055: Malformed Resource Enumeration Argument Vulnerability (CVE-1999-0980)
- MS99-057 Malformed Security Identifier Request Vulnerability (CVE-1999-0995)
- MS00-003 Spoofed LPC Port Request Vulnerability (CVE-2000-0070)
- MS00-040 Remote Registry Access Authentication Vulnerability (CVE-2000-0377)
- MS00-062 Local Security Policy Corruption Vulnerability (CVE-2000-0771)
- MS00-066 Malformed RPC Packet Vulnerability
- MS00-070 Multiple LPC and LPC Ports Vulnerabilities
- MS01-041 Malformed RPC Request Can Cause Service Failure (CVE-2001-0509)
- MS01-048 Malformed Request to RPC Endpoint Mapper can Cause RPC Service to Fail (CVE-2001-0662)
- MS03-001 Unchecked Buffer in Locator Service Could Lead to Code Execution (CVE-2003-0003)
- MS03-010 Flaw in RPC Endpoint Mapper Could Allow Denial of Service Attacks (CVE-2002-1561)
- MS03-026 Buffer Overrun in RPC Interface Could Allow Code Execution (CVE-2003-0352)
- MS03-039 Buffer Overrun in RPCSS Service Could Allow Code Execution (CVE-2003-0528, CVE-2003-0605, CVE-2003-0715)
- MS03-043 Buffer Overrun in Messenger Service Could Allow Code Execution (CVE-2003-0717)
- MS03-049 Buffer Overrun in the Workstation Service Could Allow Code Execution (CVE-2003-0812)
- MS04-011 LSASS vulnerability (CVE-2003-0533)
- MS04-012 RPC Runtime Library Vulnerability, RPCSS Service Vulnerability, COM Internet

Services (CIS) - RPC over HTTP Vulnerability, Object Identity Vulnerability (CVE-2003-0807, CVE-2003-0813, CVE-2004-0116, CVE-2004-0124)

- MS04-029 Vulnerability in RPC Runtime Library Could Allow Information Disclosure and Denial of Service (CVE-2004-0569)
- MS04-031 Vulnerability in NetDDE Could Allow Remote Code Execution (CVE-2004-0206)
- MS04-044 Windows Kernel Vulnerability (CVE-2004-0893)
- MS05-007 Vulnerability in Windows Could Allow Information Disclosure (CVE-2005-0051)
- MS05-010 Vulnerability in the License Logging Service Could Allow Code Execution (CVE-2005-0050)
- MS05-017 Vulnerability in Message Queuing Could Allow Code Execution (CVE-2005-0059)
- MS05-039 Vulnerability in Plug and Play Could Allow Remote Code Execution and Elevation of Privilege (CVE-2005-1983)
- MS05-040 Vulnerability in Telephony Service Could Allow Remote Code Execution (CVE-2005-0058)
- MS05-043 Vulnerability in Print Spooler Service Could Allow Remote Code Execution (CVE-2005-1984)
- MS05-046 Vulnerability in the Client Service for NetWare Could Allow Remote Code Execution (CVE-2005-1985)
- MS05-047 Vulnerability in Plug and Play Could Allow Remote Code Execution and Local Elevation of Privilege (CVE-2005-2120)
- MS05-051 Vulnerability in MSDTC and COM+ Could Allow Remote Code Execution (CVE-2005-2119)
- MS06-008 Vulnerability in Web Client Service Could Allow Remote Code Execution (CVE-2006-0013)
- MS06-018 Vulnerability in Microsoft Distributed Transaction Coordinator Could Allow Denial of Service (CVE-2006-0034, CVE-2006-1184)

The following table lists the MSRPC interfaces that were affected by vulnerabilities:

Table 4.135. Vulnerabilities in MSRPC interfaces

Microsoft Security Bulletin	Publication Date	Affected MSRPC interface (s)	Affected software	Reference
MS99-020	June 23, 1999	lsarpc	Windows NT 4.0	CVE-1999-0721
MS99-055	December 09, 1999	srvsvc	Windows NT 4.0	CVE-1999-0980

MS99-057	December 16, 1999	lsarpc	Windows NT 4.0	CVE-1999-0995
MS00-040	June 08, 2000	winreg	Windows NT 4.0	CVE-2000-0377
MS00-062	August 28, 2000	lsarpc	Windows 2000	CVE-2000-0771
MS01-041	July 26, 2001	Multiple interfaces	Windows NT 4.0, 2000, Exchange, SQL Server	CVE-2001-0509
MS01-048	September 10, 2001	epmp	Windows NT 4.0	CVE-2001-0662
MS03-001	January 22, 2003	locator	Windows NT 4.0, 2000, XP	CVE-2003-0003
MS03-010	March 26, 2003	epmp	Windows NT 4.0, 2000, XP	CVE-2002-1561
MS03-026	July 16, 2003	ISystemActivator, IRemoteActivation (IActivation)	Windows NT 4.0, 2000, XP, Server 2003	CVE-2003-0352
MS03-039	September 10, 2003	ISystemActivator, IRemoteActivation (IActivation)	Windows NT 4.0, 2000, XP, Server 2003	CVE-2003-0528, CVE-2003-0605, CVE-2003-0715
MS03-043	October 15, 2003	msgsvc	Windows NT 4.0, 2000, XP, Server 2003	CVE-2003-0717
MS03-049	November 11, 2003	wkssvc	Windows 2000, XP	CVE-2003-0812
MS04-011	April 13, 2004	dssetup	Windows 2000, XP	CVE-2003-0533
MS04-012	April 13, 2004	IRemoteActivation (IActivation)	Windows 2000, XP	CVE-2004-0116, CVE-2004-0124
MS04-031	October 12, 2004	nddeapi	Windows NT 4.0, 2000, XP, Server 2003	CVE-2004-0206

MS05-007	February 8. 2005	srvsvc	Windows XP	CVE-2005-0051
MS05-010	February 8, 2005	llsrpc	Windows NT 4.0, 2000, Server 2003	CVE-2005-0050
MS05-017	April 12, 2005	qmcomm	Windows 2000, XP SP1	CVE-2005-0059
MS05-039	August 9, 2005	pnp	Windows 2000, XP, Server 2003	CVE-2005-1983
MS05-040	August 9, 2005	tapsrv	Windows 2000, XP, Server 2003	CVE-2005-0058
MS05-043	August 9, 2005	spoolss	Windows 2000, XP, Server 2003	CVE-2005-1984
MS05-046	October 11, 2005	nwwks	Windows 2000, XP, Server 2003	CVE-2005-1985
MS05-047	October 11, 2005	pnp	Windows 2000, XP	CVE-2005-2120
MS05-051	October 11, 2005	IXnRemote	Windows 2000, XP, Server 2003	CVE-2005-2119
MS06-008	February 14, 2006	davclntrpc	Windows XP, Server 2003	CVE-2006-0013
MS06-018	May 9, 2006	IXnRemote	Windows 2000, XP, Server 2003	CVE-2006-0034, CVE-2006-1184

Interesting security advisories related to MSRPC:

- Multiple remote DoS vulnerabilities in Microsoft DCE/RPC deamons (Todd Sabin, July 2001)
- Windows 2000 DCOM clients may leak sensitive information onto the network (Todd Sabin, April 2002)
- DCE RPC Vulnerabilities New Attack Vectors Analysis (Javier Kohen, Juliano Rizzo, December 2003)
- Anonymous attackers can force DCOM applications into listening on the network (Todd Sabin, October 2004)

Other MSRPC-related bugs:

- Telnet to Port 135 Causes 100 Percent CPU Usage (Windows NT 4.0)

- Denial of Service in Applications Using RPC over Named Pipes (Windows NT 4.0)
- A truncated Samba server response causes an access violation in the Lsass.exe program on a Windows XP-based client computer (Windows XP)

Prev | Up | Next

4.15. RPC interfaces restriction in Windows XP SP2, Windows Server 2003 SP1 and later versions

Home

4.17. MSRPC network traffic

4.17. MSRPC network traffic

4.17.1. MSRPC network traffic analysis with Ethereal
4.17.2. MSRPC network traffic analysis in Network Intrusion Prevention Systems
4.17.3. MSRPC network traffic analysis in Firewalls

As explained in Section 4.3, “MSRPC transports”, MSRPC was designed to be transport-independant, which implies that the MSRPC network traffic will be observed over different network protocols (TCP, UDP, SMB, HTTP).

Being able to analyze MSRPC network traffic is important for several reasons:

- MSRPC is a core protocol of Windows environments and as such, it is frequent to analyze MSRPC traffic to diagnose problems.
- The numerous vulnerabilities discovered in the MSRPC subsystem (see Section 4.16, “MSRPC vulnerabilities”) pose a serious security risk for Windows infrastructures. Network security devices such as Network Intrusion Prevention/Detection Systems (NIPS/NIDS) and Firewalls must be capable of analyzing MSRPC traffic to block exploitation of MSRPC vulnerabilities.

4.17.1. MSRPC network traffic analysis with Ethereal

Ethereal is certainly the best network analyzer to analyze MSRPC traffic. The following features are supported:

- Ability to analyze MSRPC traffic over TCP, UDP, SMB and SMB2.
- Decoding of the following MSRPC interfaces: atsvc, browser, dnsserver, drsuapi, dssetup, efsrpc, eventlog, frsapi, frsrpc, InitShutdown, lsarpc, mapi, msgsvcsend, netdfs, netlogon, nspi, pnp, rras, samr, spoolss, srvsvc, svcctl, tapsrv, trksvr, winreg, wkssvc
- Ability to decrypt MSRPC traffic encrypted with Kerberos, when the appropriate keys are specified in a keytab file.
- Service response time statistics on a per-interface basis

The DCE RPC dissector of Ethereal is a heuristic dissector and will try to dissect TCP segments or UDP datagrams as DCE RPC Protocol Data Units. Sometimes, this can lead to incorrect dissections, when the dissector believes that data correspond to DCE RPC traffic when it is not the case. In that case, it is possible to force the dissection with another protocol, using the Decode As feature.

The DCE RPC dissector is able to keep track of the interface bound between a client and a server, to be able to decode appropriately the PDUs. If the network trace does not contain enough information (BIND or similar PDUs), Ethereal will stop the dissection at the DCE RPC level. In that case, it is possible to use the Decode As DCE RPC function to force the dissection as one of the DCE RPC interfaces supported by Ethereal.

For MSRPC over SMB traffic, the SMB dissector calls the DCE RPC dissector for any named pipe different from the **LANMAN** pipe. This is because the **LANMAN** pipe is used to carry RAP (Remote Administration Protocol) traffic and not DCE RPC traffic.

Some of the MSRPC dissectors are auto-generated using IDL files and two different IDL compilers, Pidl from the Samba project and idl2eth, part of Ethereal.

The dissectors involved in the dissection of MSRPC traffic handle data reassembly, so that the DCE RPC dissector can reassemble fragmented DCE RPC PDUs. For best results, the following dissector options have to be enabled:

- TCP dissector: Allow subdissector to reassemble TCP streams

- SMB dissector: Reassemble SMB Transaction payload
- SMB dissector: Reassemble DCERPC over SMB
- DCERPC dissector: Reassemble DCE/RPC messages spanning multiple TCP segments
- DCERPC dissector: Reassemble DCE/RPC fragments

4.17.2. MSRPC network traffic analysis in Network Intrusion Prevention Systems

Because of the numerous vulnerabilities discovered in MSRPC (see Section 4.16, “MSRPC vulnerabilities”), Network Intrusion Prevention and Detection Systems must inspect MSRPC traffic to detect or block malicious traffic.

Because the protocols involved (SMB, MSRPC) are complex, implementation of MSRPC traffic analysis in a network security device is a complex task that requires a good understanding of the protocols. Several evasion techniques are possible if the implementation of these protocols is not complete.

The successive improvements in NFR's MSRPC package gives a good idea of the work required to successfully implement MSRPC in NIPS:

- RAPID RESPONSE - MSRPC Version 21 (MS06-018 and Feature Upgrades): MSRPC package, version 21.
- MAINTENANCE - MSRPC Version 20: MSRPC package, version 20.
- RAPID RESPONSE - MSRPC Version 19 (MS05-047 and MS05-051): MSRPC package, version 19.
- RAPID RESPONSE: MSRPC Version 18 for MS05-039 (UPnP) exploit: MSRPC package, version 18.
- Beyond "Blaster" - MSRPC Evasions: nfr(sensor), June 2005.
- UPDATE - MSRPC Version 16: MSRPC package, version 16.
- MAINTENANCE - MSRPC Update (Version 15): MSRPC package, version 15.
- RAPID RESPONSE - MSRPC Update (v14) for MS05-007 and MS05-010 Microsoft vulnerabilities: MSRPC package, version 14.
- MAINTENANCE: Updated SMB Version 5 and MSRPC Version 13 packages available: MSRPC package, version 13.
- MAINTENANCE - MSRPC Update (Version 11) and SMB Update (Version 3): MSRPC package, version 11.
- Blasting "Blaster"-Detecting the MSRPC DCOM hole: nfr(sensor), fall 2003
- MSRPC Package Update - New Version of MSRPC (Version 10) Contains Important Bugfixes: MSRPC package, version 10.
- Important MSRPC DCOM package update - MS Messenger Service (MS03-043): MSRPC package, version 9.

- Important Note for all customers - MSRPC DCOM Rapid Response Update (Version 8): MSRPC package, version 8.
- Important Note for all customers - MSRPC DCOM Rapid Response Update: MSRPC package, version 7.
- Important Note for all customers - MSRPC DCOM Rapid Response Update: MSRPC package, version 6.
- DCOM Worm/MSRPC Update for NID-100 and NID-200 Customers: MSRPC package, version 5.
- UPDATED MS-RPC Rapid Response from NFR RRT: MSRPC package, version 4.
- UPDATED MS-RPC Rapid Response from NFR RRT: MSRPC package, version 2.
- NEW Package available to detect MSRPC DoS: MSRPC package, version 1.

4.17.3. MSRPC network traffic analysis in Firewalls

To properly implement a network security policy in Windows environments, it might be desirable to use firewalls that support MSRPC.

Depending on the completeness of the implementation, MSRPC support in a firewall might include the following features:

- Support of multiple protocols sequences: TCP, UDP, SMB, HTTP
- Enforce sanity checks to network traffic, to ensure that it corresponds to legitimate DCE RPC PDUs.
- Keep state in the MSRPC filtering engine, to match requests and responses
- Possibility to specify a list of allowed or denied interfaces (using UUIDs)
- Possibility to specify a list of allowed or denied operations for specific interfaces
- Possibility to analyze results of endpoint mapper queries and dynamically allow traffic to ports revealed in answers to endpoint mapper queries
- Possibility to analyze results of DCOM activation requests and dynamically allow traffic to ports needed to reach allowed DCOM servers
- Possibility to block SMB NULL sessions
- Possibility to restrict connections to certain named pipes (SMB transport)
- Possibility to block unauthenticated MSRPC sessions (especially when the TCP transport is used)

With Windows Server 2003 SP1, a modification to the MSRPC implementation was introduced. As a consequence, firewalls implementing sanity checks on MSRPC traffic started to block traffic originating from these systems because the software did not consider the traffic as valid. Software updates are available from the different vendors to fix the problem.

4.18. DCOM

4.18.1. COM interfaces
4.18.2. DCOM network traffic

For a good introduction to COM and DCOM, see Implementing Distributed COM in Samba, written by Jelmer Vernooij (Samba Team Member).

The Distributed Component Object Model Protocol -- DCOM/1.0 Internet Draft, published by Microsoft in January 1998, is still available.

4.18.1. COM interfaces

A process that hosts COM objects will typically support interfaces among the following ones:

```
00000001-0000-0000-c000-000000000046 v0.0 (IClassFactory)
00000131-0000-0000-c000-000000000046 v0.0 (IRemUnknown)
00000132-0000-0000-c000-000000000046 v0.0 (ILocalSystemActivator)
00000134-0000-0000-c000-000000000046 v0.0 (IRunDown)
00000143-0000-0000-c000-000000000046 v0.0 (IRemUnknown2)
```

Table 4.136. IRemUnknown methods

Interface	Method number	Method name
00000131-0000-0000-c000-000000000046 v0.0: IRemUnknown		
	0x00	QueryInterface
	0x01	AddRef
	0x02	Release
	0x03	RemQueryInterface
	0x04	RemAddRef
	0x05	RemRelease

The **IRemUnknown2** interface inherits from the **IRemUnknown** interface and adds one method, **RemQueryInterface2**.

Table 4.137. IRemUnknown2 methods

Interface	Method number	Method name
00000143-0000-0000-c000-000000000046 v0.0: IRemUnknown2		
	0x00	QueryInterface
	0x01	AddRef

	0x02	Release
	0x03	RemQueryInterface
	0x04	RemAddRef
	0x05	RemRelease
	0x06	RemQueryInterface2

Definitions of core COM interfaces can be obtained in IDL files published by the WINE project:

- unknwn.idl
- objidl.idl
- oleidl.idl
- ocidl.idl

The Oleview Microsoft tool can be used to examine and analyze registered COM interfaces on a Windows system.

Table 4.138. IOrCallback operations

Interface	Operation number	Operation name
18f70770-8e64-11cf-9af1-0020af6e72f4 v0.0: IOrCallback		
	0x00	UseProtSeq
	0x01	GetCustomProtseqInfo
	0x02	UpdateResolverBindings

4.18.2. DCOM network traffic

The Understanding the DCOM Wire Protocol by Analyzing Network Data Packets article, published in the March 1998 issue of the Microsoft Systems Journal publication, documents how DCOM is implemented at the network level.

The DCOM wire protocol uses DCE RPC as its transport protocol. Ethereal supports the DCOM wire protocol and has dissectors for the following core COM interfaces:

- IOXIDResolver
- ISystemActivator and IRemoteActivation
- IRemUnknown and IRemUnknown2
- IDispatch

When analyzing DCOM traffic with Ethereal, it is recommended to use the Windows version of Ethereal because it is able to use the Windows registry to translate IID's (GUID's) to interfaces names.

Chapter 5. Conclusion

Because of the proprietary nature of the Windows operating system, Windows network services internals have been progressively documented by independent researchers.

In the past, many vulnerabilities have been discovered in the Windows SMB and MSRPC implementations. Recently, multiple vulnerabilities in the MSRPC implementation have been published.

A good knowledge of Windows-specific network protocols is required to properly mitigate associated risks.

Bibliography

[1] Implementing CIFS: http://www.ubiqx.org/cifs/

[2] The Cable Guy TechNet column: http://www.microsoft.com/technet/community/columns/cableguy/default.mspx

[3] Windows Network Data and Packet Filtering: http://www.ndis.com/papers/winpktfilter.htm

[4] NAT Clients Cannot View Web Sites After You Install SQL 2000 SP2 or SP3 on an RRAS Server: http://support.microsoft.com/?id=324288

[5] Netstat Does Not Display Listening TCP Ports: http://support.microsoft.com/?id=131482

[6] App Request UDP Only, "Netstat -an" Displays TCP and UDP: http://support.microsoft.com/?id=194171

[7] The NETSTAT Command Incorrectly Shows Ports in Listening States: http://support.microsoft.com/?id=331078

[8] hping: http://www.hping.org/

[9] Netcat for NT 1.11: http://www.vulnwatch.org/netcat/

[10] TDIMon: http://www.sysinternals.com/Utilities/TdiMon.html

[11] Local Host Monitor 1.1: http://www.ntkernel.com/w&p.php?id=24

[12] HOW TO: Determine Which Program Uses or Blocks Specific Transmission Control Protocol Ports in Windows http://support.microsoft.com/?id=281336

[13] The netstat command can now display process IDs that correspond to active TCP or UDP connections in Windows 2000: http://support.microsoft.com/?id=907980

[14] TCPView: http://www.sysinternals.com/Utilities/TcpView.html

[15] Network Ports Used by Key Microsoft Server Products: http://www.microsoft.com/smallbusiness/support/articles/ref_net_ports_ms_prod.mspx

[16] fport: http://www.foundstone.com/knowledge/proddesc/fport.html

[17] NT port binding insecurity: http://www.insecure.org/sploits/NT.port-binding-vulnerability.html

[18] socat - Multipurpose relay: http://www.dest-unreach.org/socat/

[19] NT needs privileged ports: http://discuss.microsoft.com/SCRIPTS/WA-MSD.EXE?A2=ind9802b&L=cifs&P=738

[20] Enabling NetBT to Open IP Ports Exclusively: http://support.microsoft.com/?id=241041

[21] Applications May Be Able To "Listen" on TCP or UDP Ports: http://support.microsoft.com/?id=194431

[22] Using SO_EXCLUSIVEADDRUSE: http://msdn.microsoft.com/library/en-us/winsock/winsock/using_so_exclusiveaddruse.asp

[23] BUG: Non-administrator users cannot set the SO_EXCLUSIVEADDRUSE option on the Winsock setsockopt API call: http://support.microsoft.com/?id=870562

[24] Windows Packet Capture Library: http://www.winpcap.org/

[25] Atelier Web Ports Traffic Analyzer: http://www.atelierweb.com/pta/index.htm

[26] HOW TO: Install Microsoft Loopback Adapter in Windows 2000: http://support.microsoft.com/?id=236869

[27] SMB: The Server Message Block Protocol. http://www.ubiqx.org/cifs/SMB.html

[28] NBT: NetBIOS over TCP/IP: http://www.ubiqx.org/cifs/NetBIOS.html

[29] Samba-TNG: http://www.samba-tng.org/

[30] Direct Hosting of SMB Over TCP/IP (Q204279): http://support.microsoft.com/?id=204279

[31] NetBT and raw SMB transport: http://www.hsc.fr/ressources/presentations/sambaxp2003/slide6.html

[32] RPC: Remote Procedure Call Control Specification Version 2: http://www.ietf.org/rfc/rfc1831.txt

[33] DCE 1.1: Remote Procedure Call: http://www.opengroup.org/onlinepubs/9629399/

[34] A brief history of Windows: http://www.advogato.org/article/596.html

[35] DCE 1.1: Remote Procedure Call - Introduction to the RPC API: http://www.opengroup.org/onlinepubs/9629399/chap2.htm#tagfcjh_2

[36] WinObj: http://www.sysinternals.com/Utilities/WinObj.html

[37] RPC tools: http://www.bindview.com/Support/RAZOR/Utilities/Windows/rpctools1.0-readme.cfm

[38] PipeList: http://www.sysinternals.com/Information/TipsAndTrivia.html

[39] Filemon for Windows: http://www.sysinternals.com/Utilities/Filemon.html

[40] npfs aliases: http://www.hsc.fr/ressources/presentations/sambaxp2003/slide21.html

[41] ifids: named pipes endpoints: http://www.hsc.fr/ressources/presentations/sambaxp2003/slide24.html

[42] PipeACL tools v1.0: http://www.bindview.com/Support/RAZOR/Utilities/Windows/pipeacltools1_0.cfm

[43] Win32 Pipe Security Editor Windows NT/2000/XP: http://www.beyondlogic.org/consulting/pipesec/pipesec.htm

[44] LogonSessions v1.1: http://www.sysinternals.com/utilities/logonsessions.html

[45] You Can Use The Llsrpc Named Pipe to Add or Delete Licenses and Create New License Groups: http://support.microsoft.com/?id=815458

[46] Vulnerability in the License Logging Service Could Allow Code Execution (885834): http://www.

microsoft.com/technet/security/bulletin/ms05-010.mspx

[47] Vulnerability in Web Client Service Could Allow Remote Code Execution (911927) http://www.microsoft.com/technet/security/bulletin/ms06-008.mspx

[48] Windows 2000, Null Sessions and MSRPC: http://www.bindview.com/support/Razor/Presentations/

[49] UserInfo and UserDump tools: http://www.hammerofgod.com/HaxorCons.htm

[50] ACL tools v1.0: http://www.bindview.com/Support/RAZOR/Utilities/Windows/acltools1.0-readme.cfm

[51] Private objects security auditing (LogAnalysis mailing list): http://sisyphus.iocaine.com/pipermail/loganalysis/2003-July/002104.html

[52] The Ethereal Network Analyzer: http://www.ethereal.com/

[53] Samba 4 IDL for the lsarpc interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/lsa.idl?view=markup

[54] Samba 4 IDL for the dssetup interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/dssetup.idl?view=markup

[55] Samba 4 IDL for the samr interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/samr.idl?view=markup

[56] Samba 4 IDL for the netlogon interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/netlogon.idl?view=markup

[57] Samba 4 IDL for the eventlog interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/eventlog.idl?view=markup

[58] Samba 4 IDL for the netdfs interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/dfs.idl?view=markup

[59] Samba 4 IDL for the srvsvc interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/srvsvc.idl?view=markup

[60] Samba 4 IDL for the svcctl interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/svcctl.idl?view=markup

[61] Samba 4 IDL for the winreg interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/winreg.idl?view=markup

[62] Samba 4 IDL for the wkssvc interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/wkssvc.idl?view=markup

[63] Ethereal SVN repository: http://anonsvn.ethereal.com/viewcvs/viewcvs.py/trunk/

[64] Windows Workstation Service Remote Buffer Overflow: http://www.eeye.com/html/Research/Advisories/AD20031111.html

[65] Buffer Overrun in the Workstation Service Could Allow Code Execution (828749): http://www.microsoft.com/technet/security/bulletin/ms03-049.mspx

[66] Samba 4 IDL for the spoolss interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/spoolss.idl?view=markup

[67] Vulnerability in Print Spooler Service Could Allow Remote Code Execution (896423): http://www.microsoft.com/technet/security/bulletin/ms05-043.mspx

[68] Minimizing Windows network services: http://www.hsc.fr/ressources/breves/min_srv_res_win.en.html

[69] dcedump (part of the SPIKE toolkit): http://www.immunitysec.com/resources-freesoftware.shtml

[70] Endpoint Mapper Interface Definition: http://www.opengroup.org/onlinepubs/009629399/apdxo.htm#tagcjh_35

[71] Distributed Component Object Model Protocol -- DCOM/1.0: http://quimby.gnus.org/internet-drafts/draft-brown-dcom-v1-spec-03.txt

[72] Microsoft Debugging Tools: http://www.microsoft.com/whdc/ddk/debugging/default.mspx

[73] Understanding the DCOM Wire Protocol by Analyzing Network Data Packets: http://www.microsoft.com/msj/0398/dcom.aspx

[74] Microsoft Windows 2000 RPC DCOM Interface DOS AND Privilege Escalation Vulnerability: http://www.securiteam.com/exploits/5CP0N0KAKK.html

[75] Locator Service Buffer Overflow Vulnerability: http://www.nextgenss.com/advisories/ms-rpc-loc.txt

[76] Unchecked Buffer in Locator Service Could Lead to Code Execution (810833): http://www.microsoft.com/technet/security/bulletin/MS03-001.mspx

[77] Windows PopUP SPAM: http://www.mynetwatchman.com/kb/security/articles/popupspam/

[78] Buffer Overrun in Messenger Service Could Allow Code Execution (828035): http://www.microsoft.com/technet/security/bulletin/MS03-043.mspx

[79] Samba 4 IDL for the atsvc interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/atsvc.idl?view=markup

[80] Samba 4 IDL for the drsuapi interface: http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/drsuapi.idl?view=markup

[81] TCP ports, UDP ports, and RPC ports that are used by Message Queuing: http://support.microsoft.com/?id=178517

[82] Vulnerability in Message Queuing Could Allow Code Execution (892944): http://www.microsoft.com/technet/security/bulletin/ms05-017.mspx

[83] drsuapi MSRPC interface Ethereal dissector: http://anonsvn.ethereal.com/viewcvs/viewcvs.py/trunk/epan/dissectors/packet-dcerpc-drsuapi.c

[84] Windows Local Security Authority Service Remote Buffer Overflow: http://www.eeye.com/html/Research/Advisories/AD20040413C.html

[85] LSASS Vulnerability - CAN-2003-0533: http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

[86] Sasser Worm Technical Analysis: http://www.eeye.com/html/Research/Advisories/AD20040501.html

[87] XCCC: Exchange 2000 Windows 2000 Connectivity Through Firewalls: http://support.microsoft.com/?id=280132

[88] RPC Interfaces That Are Exposed by Secure Mail Publishing in ISA Server 2000: http://support.microsoft.com/?id=304948

[89] How MAPI Clients Access Active Directory: http://support.microsoft.com/?id=256976

[90] Be Wary of Other RPC Endpoints Running in the Same Process: http://msdn.microsoft.com/library/en-us/rpc/rpc/be_wary_of_other_rpc_endpoints_running_in_the_same_process.asp

[91] Process Explorer: http://www.sysinternals.com/Utilities/ProcessExplorer.html

[92] services.exe RPC services: http://www.hsc.fr/ressources/presentations/sambaxp2003/slide26.html

[93] DCE/RPC over SMB: Samba and Windows NT Domain Internals. Luke Kenneth Casson Leighton. Macmillan Technical Publishing, 2000.

[94] RPC Interface Restriction: Changes to Functionality in Microsoft Windows XP Service Pack 2 (Part 2: Network Protection Technologies) http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/sp2netwk.mspx#EHAA

[95] List of Remote Procedure Call (RPC) fixes in Windows XP Service Pack 2 and in Windows XP Tablet PC Edition 2005: http://support.microsoft.com/?id=838191

Prev

Chapter 5. Conclusion Home