# Services réseaux des systèmes Windows
# Etude de cas avec Windows 2000 et Windows XP

Jean-Baptiste Marchand
Jean-Baptiste.Marchand@hsc.fr


## Table des matières

# 1 Introduction

Ce document présente une classification des différents types de services réseaux sur TCP/IP typiquement rencontrés sur un système Windows. L'objectif de cette classification est de mettre en évidence le rôle de chaque service, de sorte qu'il soit ensuite possible de sécuriser un système en n'activant que les fonctionnalités réseaux strictement nécessaires.

L'approche adoptée est la suivante :

– Enumération des ports TCP et UDP ouverts

– Identification des services réseaux utilisant ces ports de communications

Les systèmes utilisés pour les expérimentations sont du type Windows 2000, version serveur et Windows XP, version professionnelle, munis d'une seule interface Ethernet, d'adresse 192.70.106.143.

# 2 Enumération des ports TCP et UDP

Le moyen de communication d'un service réseau est un (ou plusieurs) port(s) de communication, TCP ou UDP.

Une méthode pour déterminer les services réseaux en fonctionnement sur un système est de lister les ports TCP et UDP en écoute. La commande `netstat` permet notamment cela.

## 2.1 Windows 2000 Advanced Server

Sur un système Windows 2000 Avanced Server installé par défaut, les ports TCP suivants sont ouverts :

```
C:\WINNT>netstat -anp tcp

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:25             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1026           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3372           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:4983           0.0.0.0:0              LISTENING
  TCP    192.70.106.143:139     0.0.0.0:0              LISTENING
```

Pour valider la sortie de la commande `netstat`, il est préférable de réaliser un balayage de ports depuis un autre système, par exemple avec l'outil Nmap [1] :

```
jbm@garbarek ~> sudo nmap -sS 192.70.106.143 -p 1-65535
```

```
Starting nmap V. 2.54BETA25 ( www.insecure.org/nmap/ )
Interesting ports on fenetre-w2k.hsc.fr (192.70.106.143):
(The 65524 ports scanned but not shown below are in state: closed)
Port       State       Service
25/tcp     open        smtp
80/tcp     open        http
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
443/tcp    open        https
445/tcp    open        microsoft-ds
1025/tcp   open        listen
1026/tcp   open        nterm
1027/tcp   open        unknown
3372/tcp   open        unknown
4983/tcp   open        unknown


Nmap run completed -- 1 IP address (1 host up) scanned in 115 seconds
```

Le balayage de ports trouve les mêmes ports TCP que ceux identifiés sur le système local avec `netstat`. Concernant les ports UDP, les ports suivants sont rapportés comme ouverts :

```
C:\WINNT>netstat -anp udp

Active Connections

  Proto  Local Address          Foreign Address        State
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1028           *:*
  UDP    0.0.0.0:1029           *:*
  UDP    0.0.0.0:3456           *:*
  UDP    192.70.106.143:137     *:*
  UDP    192.70.106.143:138     *:*
  UDP    192.70.106.143:500     *:*
```

La sortie de `nmap` confirme ce résultat :

```
jbm@garbarek ~> sudo nmap -sU 192.70.106.143 -p 1-65535

Starting nmap V. 2.54BETA25 ( www.insecure.org/nmap/ )
Interesting ports on fenetre-w2k.hsc.fr (192.70.106.143):
(The 65527 ports scanned but not shown below are in state: closed)
Port       State       Service
135/udp    open        loc-srv
137/udp    open        netbios-ns
138/udp    open        netbios-dgm
445/udp    open        microsoft-ds
500/udp    open        isakmp
1028/udp   open        unknown
```

```
1029/udp   open        unknown
3456/udp   open        vat


Nmap run completed -- 1 IP address (1 host up) scanned in 106 seconds
```

## 2.2 Windows XP Professional

La commande `netstat` de Windows XP supporte l'option `-o`, qui affiche l'identifiant du processus ayant ouvert le port. La commande `tasklist` permet alors de retrouver le nom du processus associé à un identifiant donné.

Sur une installation par défaut de Windows XP Professional, les ports TCP en écoute sont les suivants :

```
C:\WINDOWS>netstat -anop tcp

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       884
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING       976
  TCP    0.0.0.0:5000           0.0.0.0:0              LISTENING       1160
  TCP    192.70.106.143:139     0.0.0.0:0              LISTENING       4
```

`nmap` confirme que les ports TCP suivants sont effectivement ouverts :

```
jbm@garbarek ~> sudo nmap -sS 192.70.106.143 -p 1-65535

Starting nmap V. 2.54BETA25 ( www.insecure.org/nmap/ )
Interesting ports on fenetre-xp.hsc.fr (192.70.106.143):
(The 65530 ports scanned but not shown below are in state: closed)
Port       State       Service
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
445/tcp    open        microsoft-ds
1025/tcp   open        listen
5000/tcp   open        fics


Nmap run completed -- 1 IP address (1 host up) scanned in 89 seconds
```

Les ports UDP ouverts sont les suivants :

```
C:\WINDOWS>netstat -anop udp

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  UDP    0.0.0.0:135            *:*                                    884
  UDP    0.0.0.0:445            *:*                                    4
```

```
  UDP    0.0.0.0:500            *:*                                    704
  UDP    0.0.0.0:1026           *:*                                    1112
  UDP    0.0.0.0:1027           *:*                                    976
  UDP    127.0.0.1:123          *:*                                    976
  UDP    127.0.0.1:1900         *:*                                    1160
  UDP    192.70.106.143:123     *:*                                    976
  UDP    192.70.106.143:137     *:*                                    4
  UDP    192.70.106.143:138     *:*                                    4
  UDP    192.70.106.143:1900    *:*                                    1160
```

nmap trouve les ports UDP suivants :

```
jbm@garbarek ~> sudo nmap -sU 192.70.106.143 -p 1-65535

Starting nmap V. 2.54BETA25 ( www.insecure.org/nmap/ )
Interesting ports on fenetre-xp.hsc.fr (192.70.106.143):
(The 65525 ports scanned but not shown below are in state: closed)
Port       State       Service
123/udp    open        ntp
135/udp    open        loc-srv
137/udp    open        netbios-ns
138/udp    open        netbios-dgm
445/udp    open        microsoft-ds
500/udp    open        isakmp
1026/udp   open        unknown
1027/udp   open        unknown
1900/udp   open        unknown


Nmap run completed -- 1 IP address (1 host up) scanned in 84 seconds
```

## 2.3 Particularités de la commande netstat

Historiquement, la commande `netstat` des systèmes Windows n'était pas très fiable. Les premières versions n'affichaient pas les ports TCP en écoute [2] ou rapportaient un port TCP comme étant en écoute alors que seul le port UDP de même numéro était ouvert [3]. A partir de Windows 2000, la commande `netstat` est plus fiable mais il reste un certain nombre de spécificités qu'il est nécessaire de connaître.

Dans les exemples précédents, nous avons utilisé les options suivantes de `netstat` :

- `-a` : permet d'afficher les ports en écoute.
- `-n` : empeche la résolution DNS des adresses IP.
- `-p proto` : affiche les ports ouverts pour le protocole spécifié par `proto` et peut valoir `tcp` ou `udp`, ainsi que `tcpv6` et `udpv6` lorsque la pile IPv6 est installée.
- `-o` (Windows XP) : affiche l'identifiant du processus ayant ouvert le port.

La sortie de ces commandes est triée en fonction de l'adresse locale, dans l'ordre suivant :

1. `0.0.0.0` : écoute sur toutes les interfaces (`INADDR_ANY`)

2. `127.0.0.1` : écoute sur l'interface de bouclage (*loopback*)
3. `xxx.yyy.zzz.ttt` : écoute sur l'interface d'adresse IP `xxx.yyy.zzz.ttt`

Pour chaque type d'adresse locale, les connexions sont triées dans l'ordre suivant :

1. ports ouverts (`LISTENING` pour TCP, vide pour UDP)
2. connexions TCP établies (`ESTABLISHED`)
3. connexions TCP en états transitoires (états d'ouverture ou de fermeture)

Les numéros de ports apparaissent immédiatement derrière l'adresse IP, après le caractère `:`.

La sortie de la commande `netstat` peut parfois être déroutante. Quelques spécificités du fonctionnement des sockets Windows sont à connaître, afin d'interpréter correctement son résultat.

### 2.3.1 Interfaces d'écoute

`netstat` rapporte parfois un port en écoute sur toutes les interfaces (adresse `0.0.0.0`) mais également sur une interface particulière. Dans ce cas, un paquet à destination de l'adresse IP de l'interface particulière sera reçue par la socket du service en écoute sur l'interface, plutôt que par la socket du service en écoute sur toutes les interfaces.

### 2.3.2 Affichage erroné de ports en écoute

`netstat` rapporte tout port source TCP utilisé dans une connexion établie à l'initiative de la machine locale (ouverture active) comme étant en écoute, sur toutes les interfaces (adresse `0.0.0.0`). Ceci est particulièrement trompeur et alourdit considérablement la sortie de la commande.

Dans l'exemple suivant, la machine locale a établi une connexion TCP du port source 1367 vers le port 22 d'une machine distante :

```
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
```

Contrairement aux apparences, la première ligne ne signifie pas que le port TCP `1367` de la machine locale est en écoute. Il est d'ailleurs possible de s'en assurer en tentant de s'y connecter :

```
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
```

Une tentative de connexion sur le port 1367 se voit refusée, avec un segment TCP RST, indiquant que le port est effectivement fermé.

Pour expliquer ce comportement, il faut regarder au niveau de la mise en oeuvre de l'API Winsock sur TCP/IP. Au sein de la pile réseau, l'interface TDI

permet d'accéder aux pilotes d'un protocole de niveau transport. Pour mettre en oeuvre l'API Winsock sur TCP/IP, un pilote traduit les appels de l'API Winsock en appels à l'interface TDI, adressés au pilote TCP/IP.

En regardant au niveau TDI quels sont les appels à cette interface lors de l'ouverture d'une connexion TCP vers un serveur distant, il est possible de comprendre pourquoi `netstat` rapporte que le port est en écoute alors que ce n'est pas le cas. Pour cela, nous utilisons l'outil TDImon [4].

A des fins de test, nous établissons une connexion TCP sur le port 22 d'un serveur distant et la fermons immédiatement avec la commande suivante :

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

Un extrait allégé de la sortie de TDImon lancé en parallèle révèle le fonctionnement suivant :

```
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
```

Les lignes précédentes peuvent être interprétées comme suit :

- La ligne 1 montre la requête d'ouverture d'un objet (`IRP_MJ_CREATE`) adressée au pilote TDI
- Les lignes 2 à 6 montrent la mise en place de gestionnaires (*handlers*), appelés lors de certains événements. En particulier, la ligne 4 met en place un gestionnaire pour l'événement de réception de données à destination du port TCP 1038.
- Les lignes 8 à 10 montrent l'ouverture de l'objet utilisé pour représenter la connexion TCP sortante. La commande `TDI_CONNECT`, à la ligne 10, établit effectivement la connexion, vers le port TCP 22 de la machine d'adresse 192.168.1.254.

Il apparaît donc qu'au niveau TDI, une connexion TCP sortante, réalisée à l'aide de l'API Winsock sur TCP/IP, utilise deux points de terminaison locaux :

- Un point de terminaison local pour représenter la connexion établie
- Un point de terminaison local pour recevoir les données à destination du port source utilisé dans la connexion établie

Le problème est que le second point de terminaison TDI est utilisé uniquement pour le fonctionnement interne de Winsock sur TCP/IP et ne correspond pas à une socket en écoute au niveau Winsock. Malheureusement, les fonctions permettant de récupérer la table des connexions TCP ne font pas la distinction entre un point de terminaison TDI utilisé pour un véritable serveur TCP en écoute et un point de connexion servant uniquement à recevoir les données d'une connexion sortante.

Ceci explique donc pourquoi des ports utilisés comme ports source de connexions TCP sortantes apparaissent en écoute alors qu'ils ne le sont pas.

Il faut donc se méfier de la liste des sockets dans l'état `LISTENING` et s'assurer que ces ports ne sont pas en réalité des ports sources utilisés dans une connexion TCP établie.

# 3 Identification des services réseaux

Une fois les ports ouverts identifiés, l'étape suivante est de déterminer quels sont les services en écoute sur quels ports. L'approche couramment utilisée pour cela est de chercher quel processus a ouvert quel(s) port(s) et de retrouver, à partir de l'identifiant du processus, le service correspondant.

Sur des systèmes Unix, cette tâche est relativement simple, par exemple à l'aide de l'outil lsof [6]. Sous Windows, les outils fport [7] (en ligne de commande) ou Active Ports [8] (en mode graphique) permettent de réaliser l'équivalent.

Sur le système Windows 2000, la commande `fport` donne le résultat suivant :

```
C:\WINNT> fport

FPort v1.33 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
832   inetinfo       ->  25    TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
832   inetinfo       ->  80    TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
412   svchost        ->  135   TCP   C:\WINNT\system32\svchost.exe
8     System         ->  139   TCP
832   inetinfo       ->  443   TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
8     System         ->  445   TCP
480   msdtc          ->  1025  TCP   C:\WINNT\System32\msdtc.exe
668   MSTask         ->  1026  TCP   C:\WINNT\system32\MSTask.exe
832   inetinfo       ->  1027  TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
480   msdtc          ->  3372  TCP   C:\WINNT\System32\msdtc.exe
832   inetinfo       ->  4983  TCP   C:\WINNT\System32\inetsrv\inetinfo.exe

412   svchost        ->  135   UDP   C:\WINNT\system32\svchost.exe
8     System         ->  137   UDP
8     System         ->  138   UDP
8     System         ->  445   UDP
244   lsass          ->  500   UDP   C:\WINNT\system32\lsass.exe
832   inetinfo       ->  1028  UDP   C:\WINNT\System32\inetsrv\inetinfo.exe
232   services       ->  1029  UDP   C:\WINNT\system32\services.exe
832   inetinfo       ->  3456  UDP   C:\WINNT\System32\inetsrv\inetinfo.exe
```

Le résultat de `fport` nous permet de déterminer que :

- Le processus `inetinfo.exe` a ouverts les ports TCP 25, 80, 443, 1027 et 4983 et les ports UDP 1028 et 3456
- Le processus `msdtc` a ouvert les ports TCP 1025 et 3372
- Le processus `svchost` a ouvert le port 135, en TCP et en UDP
- Le processus `MSTask` a ouvert le port TCP 1026

– Le processus `lsass` a ouvert le port UDP 500
– Le processus `services` a ouvert le port UDP 1029

Les autres ports TCP (139, 445) et UDP (137, 138 et 445) sont rapportés comme ouverts par le processus `System`.

Si nous reprenons dans l'ordre, nous voyons que le processus `inetinfo` a ouvert plusieurs ports dont les ports utilisés par les services SMTP (25), HTTP (80) et HTTPS (443). Il peut paraître surprenant de voir un même processus ouvrir plusieurs ports de nature différente. L'explication est que, sous Windows, un unique processus peut héberger plusieurs services. Ici, le processus `inetinfo` est celui qui héberge les services du serveur Internet Microsoft IIS. Dans le cas présent, les services SMTP et HTTP/HTTPS sont activés et fonctionnent tous deux dans le processus `inetinfo`.

Avec cet exemple, nous voyons qu'il ne suffit pas toujours d'identifier le processus ayant ouvert un port pour identifier le service utilisant ce port.

# 4 Classification des services réseaux

Dans cette partie, nous utilisons une méthode nous permettant de classer les services réseaux, à partir des ports ouverts observés. Nous commençons par les ports ouverts correspondants à des services Internet classiques (*well-know ports*).

## 4.1 Services Internet classiques

Les services Internet classiques sont ceux qui utilisent des ports bien connus, tels que ceux utilisés par les protocoles classiques de l'Internet tels SMTP ou HTTP. Ces ports peuvent être facilement identifiés dans la sortie de la commande `netstat` et sont souvent inférieurs à 1024.

### 4.1.1 Internet Information Server (IIS)

Windows propose en standard le serveur Internet Information Server (IIS), qui supporte les protocoles SMTP (port TCP 25), NTTP (port TCP 119), HTTP (port TCP 80) et HTTPS (port TCP 443), sous la forme de 3 services.

Si vous observez l'un de ces ports dans la sortie de la commande `netstat` et que le processus l'ayant ouvert est `inetinfo`, c'est qu'au moins un des service IIS est actif.

Sur le système Windows 2000, nous avons observé que les ports 25, 80 et 443 étaient ouverts par le processus `inetinfo`. En effet, un système Windows 2000 de type serveur installé par défaut fait fonctionner les services SMTP et HTTP d'IIS.

Nous avons également observé que le port TCP 4983 était ouvert. Par défaut, le service HTTP d'IIS installe un site d'administration, accessible sur un numéro de port alloué aléatoirement à l'installation. Dans notre installation, le port affecté est 4983, comme il est possible de s'en assurer en regardant les propriétés du site d'administration dans la console de gestion IIS.

Le port UDP 3456 est ouvert par le service d'administration IIS. Son utilité ne semble pas être connue à ce jour.

Enfin, les ports immédiatement supérieurs à 1024 attribués au processus `inetinfo` (1027 et 1028) sont des ports alloués à des services RPC, comme nous le verrons plus loin.

### 4.1.2 Internet Key Exchange (IKE)

Le protocole IKE (Internet Key Exchange) est utilisé pour l'échange de clés dans le protocole de sécurisation réseau IPsec. Il utilise le port UDP 500.

Sous Windows, un service dédié à ce protocole fonctionne dans le processus de l'autorité de sécurité locale (LSA, *Local Security Authority*). C'est pour cette raison que `fport` indique que le port UDP 500 a été ouvert par le processus `lsass`.

### 4.1.3 Network Time Protocol (NTP)

Le protocole NTP (Network Time Protocol) permet d'assurer la synchronisation horaire de systèmes via le réseau. Il utilise le port UDP 123.

Sous Windows XP, le service *Windows Time* fonctionne par défaut. Le port UDP 123 apparaît donc comme ouvert, comme nous avons pu le voir dans la sortie de la commande `netstat`.

### 4.1.4 Universal Plug and Play (UPnP)

La norme UPnP (Universal Plug and Play) est un ensemble de protocoles permettant à des systèmes et des équipements réseaux de collaborer, sans configuration préalable. Cette norme n'est supportée que sur les systèmes Windows XP et ultérieurs, sous la forme d'un service nommé *SSDP Discovery Service*. Il écoute sur le port TCP 5000 et le port UDP 1900.

Trois vulnérabilités graves [9] ont été découvertes dans ce service, il est donc vivement conseillé de le désactiver.

### 4.1.5 Dynamic Host Configuration Protocol (DHCP)

Le protocole DHCP (Dynamic Host Configuration Protocol) permet à des systèmes de récupérer de façon dynamique des paramètres de configuration IP. Ce protocole utilise le port UDP 68.

Nous n'avons pas vu de port UDP 68 ouvert dans la sortie de `netstat` car les systèmes de test ont une adresse IP statique. Si DHCP est utilisé sur un système, le port UDP 68 apparaîtra comme ouvert. C'est le cas d'un système installé complètement par défaut.

### 4.1.6 Domain Name Service (DNS)

Le protocole DNS (Domain Name Service) est utilisé pour la résolution de noms en adresses IP et vice-versa. Il utilise des datagrammes UDP, à destination du port 53.

Sur les systèmes Windows 2000 et Windows XP, un service de cache des réponses à des requêtes DNS fonctionne par défaut. Sous Windows 2000, les datagrammes UDP contenant des requêtes DNS sont envoyées avec un port source UDP alloué dynamiquement à chaque requête. Sous Windows XP, le

port source est toujours le même et est alloué une seule fois, à la première requête faite par le service.

Sur le système Windows XP, la commande `netstat` a rapporté que le port UDP 1026 était ouvert. Il s'agit justement du port source que s'est vu alloué le service de cache DNS pour ses requêtes.

Les tests suivants permettent de vérifier que c'est effectivement ce port qui est utilisé.

```
C:\WINDOWS> ping -n 1 www.google.com
```

`ping` peut être remplacé par toute autre commande demandant une résolution DNS mais pas par `nslookup`, qui gère ses propres résolutions sans passer par le service de cache DNS.

Avec un analyseur réseau, nous observons :

```
jbm@garbarek ~> sudo tcpdump -ni ep1 host 192.70.106.143

192.70.106.143.1026 > 192.70.106.99.53:  4+ A? www.google.com.  (32)
                    ^^^^
192.70.106.99.53 > 192.70.106.143.1026:  4 1/4/4 A 216.239.39.101 (184)
192.70.106.143 > 216.239.39.101: icmp: echo request
```

Le port source utilisé est donc bien 1026. Le même comportement est observé avec une autre requête :

```
C:\WINDOWS> ping -n 1 www.hsc.fr

jbm@garbarek ~> sudo tcpdump -ni ep1 host 192.70.106.143

192.70.106.143.1026 > 192.70.106.99.53:  5+ A? www.hsc.fr. (28)
                    ^^^^
192.70.106.99.53 > 192.70.106.143.1026:  5* 2/2/2 CNAME itesec.hsc.fr., (137)
192.70.106.143 > 192.70.106.33: icmp: echo request
```

Le port 1026 est donc utilisé pour l'émission de toutes les requêtes DNS. Ce port reste donc ouvert et apparaît dans la sortie de la commande `netstat`.

## 4.2 NetBIOS sur TCP/IP

NetBIOS est un protocole de niveau session utilisé très largement dans les réseaux de systèmes Windows. Il est transporté dans un protocole de transport, typiquement NetBT (NetBIOS sur TCP/IP) ou NetBEUI. Jusqu'à Windows 2000, NetBIOS sur TCP/IP était le protocole typiquement utilisé pour le transport du protocole CIFS.

Sachant que le protocole CIFS (Common Internet File System) est le protocole sur lequel repose les fonctionnalités de partage de ressources en environnement Windows (partage de fichiers et d'imprimantes), ainsi que les fonctionnalités d'administration à distance, il apparait évident que NetBIOS sur TCP/IP est un protocole omniprésent, du moins jusqu'à Windows 2000.

A partir de Windows 2000, le protocole CIFS peut être transporté directement dans TCP/IP, via le port 445, en supprimant la couche NetBIOS.

NetBIOS sur TCP/IP utilisent 3 ports :

– 137/UDP pour le résolution de noms NetBIOS (en diffusion ou via un serveur WINS)
– 138/UDP pour NetBIOS sans session
– 139/TCP pour NetBIOS avec session

Ces 3 ports se retrouvent dans les sorties de la commande `netstat` vues précédemment, puisque NetBIOS sur TCP/IP est activé par défaut, aussi bien sur Windows 2000 que sur Windows XP.

## 4.3 CIFS/SMB

Le protocole CIFS [10] (également connu sous le nom de ses commandes, SMB, Server Message Block) est au coeur des fonctionnalités de partage de ressources (partage de fichiers et d'imprimantes). Comme vu précédemment, il peut être transporté sur NetBIOS sur TCP/IP (port TCP 139) ou directement sur TCP/IP (port TCP 445).

Au niveau système [11], le protocole CIFS est mis en oeuvre avec une partie cliente et une partie serveur, fonctionnant chacun sous la forme d'un service :
– le service *workstation* pour le côté client
– le service *server* pour le côté serveur

Les commandes `net config workstation` et `net config server` permettent d'afficher les transports que ces services peuvent utiliser :

```
C:\WINDOWS> net config workstation

Computer name                        \\JAMAL
Full Computer name                   jamal.domus
User name                            jbm

Workstation active on
        NetbiosSmb (000000000000)
        NetBT_Tcpip_{AB128BC0-4EA0-40B1-801A-D324891BD60F} (525405FDC5F9)

Software version                     Windows 2002

Workstation domain                   WORKGROUP
Workstation Domain DNS Name          (null)
Logon domain                         JAMAL

COM Open Timeout (sec)               0
COM Send Count (byte)                16
COM Send Timeout (msec)              250
The command completed successfully.
```

Les transports possibles apparaissent sous la ligne `Workstation active on` :
– `NetbiosSmb` désigne le transport de CIFS dans TCP directement (port TCP 445, sans couche NetBIOS) [12].
– `NetBT_Tcpip_{...}` désigne le transport de CIFS dans NetBIOS sur TCP/IP, sur l'un des adaptateurs réseau.

CIFS est un protocole relativement complexe, ayant connu de nombreuses évolutions qui ont abouti à plusieurs versions du protocole, appelées dialectes.

La complexité du protocole, la faiblesse des protocoles d'authentification utilisés ainsi que des erreurs d'implémentations ont été la cause d'un certain nombre de vulnérabilités dans les systèmes Windows [13].

## 4.4 MSRPC

### 4.4.1 Introduction

Le mécanisme des RPC (Remote Procedure Call) permet à une application d'invoquer des procédures distantes (i.e. exécutée sur un système distant) de façon relativement transparente, comme si ces procédures étaient locales.

Il existe plusieurs variantes du mécanisme des RPC. Les systèmes Windows utilisent les RPC issus de la norme DCE (Distributing Computing Environment), dont les spécifications sont librement disponibles [14].

Dans les systèmes Windows, les deux principaux cas d'utilisation du mécanisme des RPC sont les suivants :

- Administration distante
- Applications distribuées, via l'infrastructure Distributed COM (DCOM)

### 4.4.2 Notion d'interface

Une interface regroupe un ensemble d'opérations (procédures) qui peuvent être appelées à distance. Chaque interface se distingue par un identifiant d'interface (*interface identifier*, abbrégé en *ifid*).

Une même interface peut être disponible en plusieurs versions : un numéro de version est donc associée à chaque mise en oeuvre d'une interface donnée.

Les procédures distantes offertes par une interface sont regroupées en vecteurs de points d'entrées (*entry point vector, EPV*). Lors de l'enregistrement d'un service RPC, une correspondance est définie entre, d'une part un identifiant d'interface et un type d'objet (repéré par un identifiant universel unique, *UUID, Universal Unique Identifier*) et, d'autre part, un vecteur de points d'entrées.

La liste des objets et de leurs types, tous deux repérés par des UUID, est définie au démarrage d'un service RPC. Ainsi, lors de l'invocation d'un service RPC contenant un identifiant d'interface et l'identifiant d'un objet donné, il est possible de trouver le type de l'objet demandé et donc le vecteur de points d'entrée, permettant d'invoquer effectivement les procédures.

Pour une explication détaillée de la notion d'interface, se reporter à la figure 2.2 de la norme DCE RPC 1.1 sus-citée.

### 4.4.3 Transport

Le mécanisme des RPC a été conçu afin d'être indépendant des protocoles de transport utilisé pour transférer les paramètres et les résultats d'appels de procédures distantes. Chaque séquence de protocoles de transport est identifiée par une chaîne de caractères humainement lisible. Dans le cas des systèmes Windows, les séquences de protocoles typiquement rencontrées sont les suivantes :

- `ncacn_ip_tcp` : transport sur TCP/IP
- `ncadg_ip_udp` : transport sur UDP/IP
- `ncacn_np` : transport sur des tubes nommés (*named pipes*)
- `ncalrpc` : RPC locaux (*local rpc*)
- `ncacn_http` : transport dans HTTP (via un serveur IIS)

Le point de terminaison (*endpoint*) d'un service RPC désigne l'attache propre au protocole de transport à partir duquel il est possible d'invoquer le service. Par exemple, pour un service RPC sur TCP/IP, le point de terminaison sera un port TCP ou UDP. Avant de pouvoir utiliser un service RPC, il faut s'attacher à un point de terminaison donné.

### 4.4.4 Enregistrement

Lorsqu'un service RPC démarre, il peut enregistrer les points de terminaison sur lesquels une interface d'identifiant donné (gérant éventuellement un type d'objet donné) est accessible.

Ainsi, lorsqu'un système distant cherche à utiliser une interface donnée, gérant éventuellement un objet d'identifiant donné, il peut interroger une base de correspondance dans laquelle se sont enregistrés les services RPC disponibles et découvrir le point de terminaison sur lequel le service peut être joint.

Ce service de correspondance est communément appelé le *portmapper*. C'est lui même un service RPC, disponible sur plusieurs protocoles de transport :

- Sur TCP, via le port 135
- Sur UDP, via le port 135
- Sur tubes nommés, via le tube `\pipe\epmapper`
- Sur HTTP, via le port TCP 593

### 4.4.5 Services RPC sur TCP/IP

Par défaut, un système Windows fait fonctionner un certain nombre de services RPC sur TCP/IP. Ces services se voient alloués des ports immédiatement supérieurs à 1024. Pour identifier de façon fiable les ports utilisés, il est possible d'interroger le *portmapper* et voir quels sont les services RPC enregistrés sur TCP/IP.

Pour celà, nous utilisons l'outil `rpcdump` [15], dont les sources sont librement disponibles. Sous un système Unix distant, il est possible d'utiliser l'outil `dcetest` [16].

Les informations rapportés par la commande `rpcdump` sont les suivantes :

- identifiant de l'interface (`IfId:`) et version (`version majeur.mineur`)
- identifiant du type des objets gérés par l'interface (`UUID:`)
- attache (`Binding:`), sous la forme `protocoles:point_de_terminaison`.
- identification lisible du service (`Annotation:`)

Sur le serveur Windows 2000, l'interrogation avec `rpcdump` du *portmapper* écoutant sur le port TCP 135 donne le résultat suivant :

```
C:\WINNT>rpcdump -p ncacn_ip_tcp 127.0.0.1

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncadg_ip_udp:192.70.106.143[1029]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: d7925959-0f67-4149-ba27-641cfe1fdbbc
```

```
Binding: ncalrpc:[LRPC000001f8.00000001]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: d7925959-0f67-4149-ba27-641cfe1fdbbc
Binding: ncacn_ip_tcp:192.70.106.143[1025]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: 061a8227-79b5-48a7-af3c-18b8908f5ac5
Binding: ncalrpc:[LRPC000001f8.00000001]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: 061a8227-79b5-48a7-af3c-18b8908f5ac5
Binding: ncacn_ip_tcp:192.70.106.143[1025]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: 5e809640-b4fa-4919-aa5f-c492e8a8b556
Binding: ncalrpc:[LRPC000001f8.00000001]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: 5e809640-b4fa-4919-aa5f-c492e8a8b556
Binding: ncacn_ip_tcp:192.70.106.143[1025]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: 39f6489f-8aa6-4195-a398-7c67bd83c077
Binding: ncalrpc:[LRPC000001f8.00000001]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: 39f6489f-8aa6-4195-a398-7c67bd83c077
Binding: ncacn_ip_tcp:192.70.106.143[1025]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[LRPC000002cc.00000001]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:192.70.106.143[1026]

IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
```

```
Binding: ncalrpc:[LRPC000002cc.00000001]

IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:192.70.106.143[1026]

IfId: 82ad4280-036b-11cf-972c-00aa006887b0 version 2.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[OLE4]

IfId: 82ad4280-036b-11cf-972c-00aa006887b0 version 2.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[INETINFO_LPC]

IfId: 82ad4280-036b-11cf-972c-00aa006887b0 version 2.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:192.70.106.143[1027]

IfId: 82ad4280-036b-11cf-972c-00aa006887b0 version 2.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\INETINFO]

IfId: 8cfb5d70-31a4-11cf-a7d8-00805f48a135 version 3.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[OLE4]

IfId: 8cfb5d70-31a4-11cf-a7d8-00805f48a135 version 3.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[INETINFO_LPC]

IfId: 8cfb5d70-31a4-11cf-a7d8-00805f48a135 version 3.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:192.70.106.143[1027]

IfId: 8cfb5d70-31a4-11cf-a7d8-00805f48a135 version 3.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\INETINFO]

IfId: 8cfb5d70-31a4-11cf-a7d8-00805f48a135 version 3.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
```

```
Binding: ncalrpc:[SMTPSVC_LPC]

IfId: 8cfb5d70-31a4-11cf-a7d8-00805f48a135 version 3.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\SMTPSVC]

IfId: bfa951d1-2f0e-11d3-bfd1-00c04fa3490a version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[OLE4]

IfId: bfa951d1-2f0e-11d3-bfd1-00c04fa3490a version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[INETINFO_LPC]

IfId: bfa951d1-2f0e-11d3-bfd1-00c04fa3490a version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:192.70.106.143[1027]

IfId: bfa951d1-2f0e-11d3-bfd1-00c04fa3490a version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\INETINFO]

IfId: bfa951d1-2f0e-11d3-bfd1-00c04fa3490a version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[SMTPSVC_LPC]

IfId: bfa951d1-2f0e-11d3-bfd1-00c04fa3490a version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\SMTPSVC]

IfId: bfa951d1-2f0e-11d3-bfd1-00c04fa3490a version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncadg_ip_udp:192.70.106.143[1028]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[ntsvcs]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
```

```
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\ntsvcs]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\scerpc]
```

Les ports immédiatement supérieurs à 1024 observés précédemment dans la sortie de la commande `netstat` se retrouvent dans la sortie de `rpcdump`, aux lignes suivantes :

- Ports TCP
  ```
  Binding: ncacn_ip_tcp:192.70.106.143[1025]
  Binding: ncacn_ip_tcp:192.70.106.143[1026]
  Binding: ncacn_ip_tcp:192.70.106.143[1027]
  ```
- Ports UDP
  ```
  Binding: ncadg_ip_udp:192.70.106.143[1028]
  Binding: ncadg_ip_udp:192.70.106.143[1029]
  ```

Ainsi, nous sommes assurés que ce sont bien des services RPC qui utilisent ces ports.

Sur le système Windows XP, la commande `rpcdump` rapporte les services RPC sur TCP/IP suivants :

```
C:\WINDOWS>rpcdump -p ncacn_ip_tcp 127.0.0.1

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncadg_ip_udp:0.0.0.0[1027]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\AudioSrv]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\pipe\\tapsrv]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[tapsrvlpc]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[wzcsvc]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
```

```
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[OLE3]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:0.0.0.0[1025]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\atsvc]

IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\AudioSrv]

IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\pipe\\tapsrv]

IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[tapsrvlpc]

IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[wzcsvc]

IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[OLE3]

IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:0.0.0.0[1025]

IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\atsvc]

IfId: 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 version 1.0
```

```
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\AudioSrv]

IfId: 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\pipe\\tapsrv]

IfId: 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[tapsrvlpc]

IfId: 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[wzcsvc]

IfId: 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[OLE3]

IfId: 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:0.0.0.0[1025]

IfId: 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\atsvc]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\AudioSrv]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\pipe\\tapsrv]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[tapsrvlpc]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
```

```
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[wzcsvc]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[OLE3]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:0.0.0.0[1025]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\atsvc]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\wkssvc]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\pipe\\keysvc]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[keysvc]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\W32TIME]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\pipe\\trkwks]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[trkwks]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
```

```
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\SECLOGON]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\srvsvc]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[senssvc]

IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\msgsvc]
```

Comme sous Windows 2000, les ports immédiatement supérieurs à 1024 se retrouvent dans la sortie de `rpcdump`, aux lignes suivantes :

- Ports TCP
  `Binding: ncacn_ip_tcp:0.0.0.0[1025]`
- Ports UDP
  `Binding: ncadg_ip_udp:0.0.0.0[1027]`

Pour terminer, précisons que des services RPC de type `ncalrpc` apparaissent au fur et à mesure de l'utilisation du système. Cependant, ces services ne sont, par définition, pas accessibles via le réseau.

### 4.4.6 Identifiants d'interface

Les identifiants d'interface, associés éventuellement à un identifiant de type, définissent les services RPC. Dans la sortie de `rpcdump` sur Windows 2000, nous avons vu les identifiants d'interface suivants :

```
IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
IfId: 82ad4280-036b-11cf-972c-00aa006887b0 version 2.0
IfId: 8cfb5d70-31a4-11cf-a7d8-00805f48a135 version 3.0
IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
IfId: bfa951d1-2f0e-11d3-bfd1-00c04fa3490a version 1.0
```

Sous Windows XP, les identifiants d'interface suivants sont apparus :

```
IfId: 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 version 1.0
IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
```

La plupart des identifiants d'interface ne sont pas documentés. Il est donc difficile de savoir exactement quelles fonctionnalités sont offertes par une interface d'identifiant donné.

En se basant sur les noms des tubes nommés, les noms des ports de communication LPC et sur un recoupement entre les ports ouverts et les processus, nous arrivons à identifier que :

- L'interface d'identifiant `5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc` est liée au service *Messenger*
- Les interfaces d'identifiants `82ad4280-036b-11cf-972c-00aa006887b0` et `8cfb5d70-31a4-11cf-a7d8-00805f48a135` et `bfa951d1-2f0e-11d3-bfd1-00c04fa3490a` sont liées à IIS.
- L'interface d'identifiant `906b0ce0-c70b-1067-b317-00dd010662da` est liée au service *Distributed Transaction Coordinator*

Les identifiants restants ne peuvent pas être identifiés avec certitude :

```
IfId: 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 version 1.0
IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
IfId: 378e52b0-c0a9-11cf-822d-00aa0051e40f version 1.0
```

### 4.4.7 Interfaces gérant un type d'objet

Dans la sortie de `rpcdump`, la ligne commencant par `UUID:` indique le type d'objets que l'interface sait gérer. Deux types de valeurs peuvent être observées :

- Identifiant de type nul (`00000000-0000-0000-0000-000000000000`)
- Identifiant de type non-nul

Dans le premier cas, l'interface pourra gérer toute requête ne précisant pas d'identifiant d'objet dans sa requête. Dans le second cas, l'interface pourra gérer toute requête spécifiant un objet dont le type est égal à l'identifiant de type enregistré, la définition du type d'un objet ayant lieu au démarrage du service RPC.

Sur le système Windows 2000, le service *Distributed Transaction Coordinator* fonctionne. Une même interface, d'identifiant `906b0ce0-c70b-1067-b317-00dd010662da` est alors enregistrée avec 4 identifiants de type différents :

```
IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: d7925959-0f67-4149-ba27-641cfe1fdbbc
Binding: ncacn_ip_tcp:192.70.106.143[1025]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: 061a8227-79b5-48a7-af3c-18b8908f5ac5
Binding: ncacn_ip_tcp:192.70.106.143[1025]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: 5e809640-b4fa-4919-aa5f-c492e8a8b556
Binding: ncacn_ip_tcp:192.70.106.143[1025]

IfId: 906b0ce0-c70b-1067-b317-00dd010662da version 1.0
Annotation:
UUID: 39f6489f-8aa6-4195-a398-7c67bd83c077
Binding: ncacn_ip_tcp:192.70.106.143[1025]
```

Ces 4 identifiants apparaissent dans la base de registres, sous `HKEY_CLASSES_ROOT\CID\` et semblent correspondre à 4 composants du service *Distributed Transaction Coordinator*, comme le laisse penser le contenu de la valeur `Description` sous chaque sous-clé de `HKEY_CLASSES_ROOT\CID` :

- MSDTC, qui est probablement le composant principal
- MSDTCXATM, qui désigne probablement le composant qui gère les transactions de type XATM (*X/Open XA Transation Manager*)
- MSDTCTIPGW, qui désigne probablement le composant qui gère les transactions TIP (*Transaction Internet Protocol*), qui utilise le port TCP 3372
- MSDTCUIS, qui désigne probablement le composant d'administration, y compris d'administration distante

Notons qu'à partir de Windows XP, il est possible de désactiver les services réseaux du *Distributed Transaction Coordinator*, de sorte que le service :

- n'enregistre pas de services RPC sur TCP/IP
- n'écoute plus sur le port TCP 3372

### 4.4.8 Services RPC sur tubes nommés

Les RPC sur tubes nommés [17] sont manipulés via le protocole CIFS. Les systèmes Windows utilisent fréquemment ce mode de transport, notamment pour les RPC d'administration distante.

Dans la sortie de `rpcdump`, quelques services RPC sur tubes nommés apparaissent, sous la forme de la séquence de protocoles de transport `ncacn_np`.

Sur le système Windows 2000, les tubes nommés suivants sont référencés :

```
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\INETINFO]
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\SMTPSVC]
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\ntsvcs]
Binding: ncacn_np:\\\\FENETRE-2K-DFLT[\\PIPE\\scerpc]
```

Sur le système Windows XP, les tubes nommés référencés sont plus nombreux :

```
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\AudioSrv]
Binding: ncacn_np:\\\\XP-DEFAULT[\\pipe\\tapsrv]
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\atsvc]
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\wkssvc]
Binding: ncacn_np:\\\\XP-DEFAULT[\\pipe\\keysvc]
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\W32TIME]
Binding: ncacn_np:\\\\XP-DEFAULT[\\pipe\\trkwks]
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\SECLOGON]
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\srvsvc]
Binding: ncacn_np:\\\\XP-DEFAULT[\\PIPE\\msgsvc]
```

Cependant, tous les tubes nommés servant de points de terminaison de services RPC n'apparaissent pas dans la sortie de `rpcdump`. Par exemple, sur un système Windows 2000, le service d'édition à distance de la base de registres (*Remote registry service*) ouvre le tube nommé `\pipe\winreg`, sur lequel un service RPC permettant l'édition de la base de registres est accessible. Ce service ne s'enregistre cependant pas dans le *portmapper*.

Le fait que la plupart des services RPC sur tubes nommés ne s'enregistrent pas auprès du *portmapper* est normal. En effet, contrairement à un service sur TCP/IP dont le numéro de port n'est pas fixe, les noms des tubes nommés sont

toujours connus. Il est donc possible d'atteindre un service RPC en se connectant directement à un tube de nom donné, sans passer par l'étape d'interrogation du *portmapper*

En examinant attentivement les sorties de `rpcdump`, il apparaît que les services RPC dont un des transports possibles est `ncacn_np` sont aussi joignables sur TCP/IP. Ceci s'explique par le fait que les services enregistrent de façon globale tous les transports sur lequel ils écoutent, y compris les transports sur des tubes nommés, même si cela n'est pas utile dans ce dernier cas.

Pour découvrir tous les tubes nommés permettant l'invocation de services RPC, il est possible de lister tous les tubes nommés ouverts sur le système et de regarder sur chacun s'il héberge des services RPC, en l'interrogeant sur les interfaces qu'il supporte.

Le programme `pipelist` [18] permet de lister tous les tubes nommés ouverts sur un système. Pour déterminer si un tube donné héberge des services RPC, il est par exemple possible d'utiliser le programme `ifids` [15] pour découvrir les éventuelles interfaces supportées via un tube donné. Si ce programme provoque une erreur ou ne rend pas la main, c'est que le tube interrogé n'héberge pas de services RPC.

Sous Windows XP, `pipelist` donne le résultat suivant :

```
C:\WINDOWS> pipelist

PipeList v1.01
by Mark Russinovich
http://www.sysinternals.com

Pipe Name                          Instances       Max Instances
---------                          ---------       -------------
TerminalServer\AutoReconnect             1               1
InitShutdown                             2              -1
SfcApi                                   2              -1
lsass                                    3              -1
protected_storage                        2              -1
ntsvcs                                  18              -1
scerpc                                   2              -1
net\NtControlPipe1                       1               1
net\NtControlPipe2                       1               1
Winsock2\CatalogChangeListener-378-0     1               1
net\NtControlPipe3                       1               1
net\NtControlPipe0                       1               1
ProfMapApi                               2              -1
winlogonrpc                              2              -1
net\NtControlPipe4                       1               1
DhcpClient                               1              -1
net\NtControlPipe5                       1               1
Winsock2\CatalogChangeListener-3d8-0     1               1
atsvc                                    2              -1
epmapper                                 2              -1
net\NtControlPipe6                       1               1
spoolss                                  2              -1
```

```
AudioSrv                                3               -1
wkssvc                                  3               -1
DAV RPC SERVICE                         2               -1
winreg                                  2               -1
ipsec                                   2               -1
keysvc                                  2               -1
WMDMPMSPpipe                            1               -1
W32TIME                                 2               -1
trkwks                                  2               -1
SECLOGON                                2               -1
PCHHangRepExecPipe                      1                8
PCHFaultRepExecPipe                     1                8
srvsvc                                  3               -1
msgsvc                                  2               -1
browser                                 2               -1
Ctx_WinStation_API_service              2               -1
ssdpsrv                                 3               -1
PIPE_EVENTROOT\CIMV2SCM EVENT PROVIDER  2               -1
net\NtControlPipe7                      1                1
```

Parmi ces tubes nommés, nous retrouvons par exemple `\pipe\epmapper`, qui héberge le service RPC *portmapper* ainsi que le gestionnaire de services COM, sur lequel nous revenons dans la partie suivante.

Avec `ifids`, il est alors possible d'énumérer les interfaces et donc les services RPC hébergés sur ce tube :

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

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
```

Le tube `\pipe\epmapper` offre donc 11 interfaces. Parmi les interfaces supportées, la première, `e1af8308-5d1f-11c9-91a4-08002b14a0fa` est celle du service de *portmapper*.

En interrogeant chaque tube nommé avec `ifids`, il apparaît qu'un certain nombre d'entre eux hébergent un nombre variable de services RPC.

Au niveau système, l'accès aux tubes nommés a lieu via le partage spécial IPC$. L'accès à ce partage se fait par le protocole CIFS, encapsulé dans NetBIOS (port TCP 139) ou directement dans TCP/IP (port TCP 445).

Il faut donc garder à l'esprit que si ces ports sont ouverts, il est possible de réaliser des RPC, notamment d'administration distante. L'exemple le plus connu est celui des sessions nulles, qui permettent de récupérer diverses informations sur un système distant, sans même s'y authentifier [19]. L'outil rpcclient [20] du projet Samba TNG est une preuve de concept des fonctions d'administration distante des systèmes Windows.

## 4.5 Distributed COM (DCOM)

Distributed COM est le terme couramment utilisé pour mettre l'accent sur le caractère distribué du modèle COM (Component Objet Model). En effet, COM permet à des applications d'utiliser de façon dynamique des composants logiciels, localisés sur le système local ou un système distant.

Dans le cas où les composants COM sont accessibles sur un serveur distant, COM utilise un protocole de communication, souvent appelé DCOM, pour utiliser les services d'un composant distant. DCOM est transporté dans DCE/RPC et est également appelé ORPC, pour Object RPC.

Pour fonctionner, DCOM utilise le port 135, qui, en plus d'héberger le service classique de portmapper RPC, contient le gestionnaire de contrôle de services COM (COM SCM).

Si nous observons la liste des services RPC accessibles sur le port 135 à l'aide de `ifids` sous Windows 2000, nous obtenons la liste suivante :

```
C:\WINNT> 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
  975201b0-59ca-11cf-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
```

Nous retrouvons exactement les mêmes interfaces que sur le tube nommé `\pipe\epmapper`.

Sous Windows XP, la liste est pratiquement identique, à l'exception de l'identifiant d'interface en troisième position :

```
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
```

Le nombre important d'interfaces tient justement au fait que le port 135 sert au fonctionnement de DCOM. Parmi ces interfaces, certaines sont des interfaces RPC classiques et d'autres sont des interfaces ORPC.

A ce jour, seuls les noms d'interfaces suivants semblent être connus :

| Identifiant | Nom | Type |
|---|---|---|
| e1af8308-5d1f-11c9-91a4-08002b14a0fa v3.0 | Portmapper | RPC |
| 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57 v0.0 | IRemoteActivation | RPC |
| 99fcfec4-5260-101b-bbcb-00aa0021347a v0.0 | IOXIDResolver | RPC |
| 00000136-0000-0000-c000-000000000046 v0.0 | ISCMLocalActivator | ORPC |
| 000001a0-0000-0000-c000-000000000046 v0.0 | ISystemActivator | ORPC |

L'usage typique des interfaces suivantes sont :

- Portmapper : service donnant l'emplacement où il est possible d'atteindre un service RPC repéré par un identifiant d'interface (typiquement un numéro de port TCP ou UDP)
- IRemoteActivation : service permettant d'instancier des objets COM à distance, afin d'en utiliser les services, via les interfaces qu'ils offrent
- IOXIDResolver : service de résolution des OXID, utilisés dans les références à des objets COM

Les noms ou l'utilité des autres interfaces restent inconnus à ce jour. Ces interfaces sont très probablement utilisées pour le fonctionnement interne de DCOM.

# 5 Bilan

Les deux tableaux, figures 1 et 2, récapitulent les services réseaux identifiés sur un système Windows 2000 serveur et Windows XP professionnel par défaut. `>1024/tcp` et `>1024/udp` signifient que le port correspondant est un port dynamiquement alloué à un service RPC.

# 6 Conclusion

Nous avons vu qu'en partant de la liste des ports ouverts sur un système Windows, nous retrouvons les services réseaux typiquement utilisés par ceux-ci. Ces services peuvent se répartir en trois grandes catégories :

- Services Internet classiques. Ces services fonctionnent sur des ports bien connus et hébergent des services classiques
- Service de partages de ressources, via le protocole CIFS/SMB, transporté ou non dans NetBIOS sur TCP/IP
- Services RPC, accessibles via différents protocoles de transport (typiquement sur TCP/IP ou tubes nommés accédés via CIFS/SMB) et utilisés

| Service | Port(s) | Service(s) Windows |
|---|---|---|
| Serveur IIS | 25/tcp | *SMTP (Simple Mail Transfert Protocol)* |
| | 80, 443/tcp | *World Wide Web Publishing Service* |
| | 3456/udp | *IIS Admin Service* |
| | >1024/tcp, >1024/udp | 2 ports dynamiques (RPC) |
| | <4000-5000>/tcp | 1 port (site d'administration) |
| Protocole IKE | 500/udp | *IPSEC Policy Agent* |
| NetBIOS sur TCP | 137/udp, 138/udp, 139/tcp | *TCP/IP NetBIOS Helper Service* |
| CIFS/SMB | 139/tcp, 445/tcp | *Server, Workstation* |
| *Portmapper* RPC | 135/tcp, 135/udp | *Remote Procedure Call (RPC)* |
| MSDTC | 3372/tcp, >1024/tcp | *Distributed Transaction Coordinator* |
| Tâches programmées | >1024/tcp | *Task Scheduler* |
| Messenger | >1024/udp | *Messenger* |

FIG. 1 – Services réseaux sur un système Windows 2000 serveur

| Service | Port(s) | Service(s) Windows |
|---|---|---|
| Protocole IKE | 500/udp | *IPSEC Policy Agent* |
| Protocole NTP | 123/udp | *Windows Time* |
| UPnP | 5000/tcp, 1900/udp | *SSDP Discovery Service* |
| Cache DNS | >1024/udp | *DNS Client* |
| NetBIOS sur TCP | 137/udp, 138/udp, 139/tcp | *TCP/IP NetBIOS Helper Service* |
| CIFS/SMB | 139/tcp, 445/tcp | *Server, Workstation* |
| *Portmapper* RPC | 135/tcp, 135/udp | *Remote Procedure Call (RPC)* |
| Tâches programmées | >1024/tcp | *Task Scheduler* |
| Messenger | >1024/udp | *Messenger* |

FIG. 2 – Services réseaux sur un système Windows XP professionnel

aussi bien pour l'administration distante que pour le fonctionnement d'applications distribuées

Pour sécuriser un système Windows au niveau réseau, il est nécessaire de ne conserver, parmi la liste des services présentés ici, uniquement ceux qui sont strictement nécessaires au fonctionnement du système.

# Références


[1] Nmap (Network Mapper) : http://www.insecure.org/nmap/

[2] Netstat Does Not Display Listening TCP Ports (Q131482) http://support.microsoft.com/support/kb/articles/Q131/4/82.ASP

[3] App Request UDP Only, "Netstat -an" Displays TCP and UDP http://support.microsoft.com/support/kb/articles/Q194/1/71.ASP

[4] TDImon : http://wwww.sysinternals.com/ntw2k/freeware/tdimon.shtml

[5] hping : http://www.hping.org/

[6] lsof : ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/

[7] fport : http://www.foundstone.com/knowledge/proddesc/fport.html

[8] Active Ports : http://www.protect-me.com/freeware.html

[9] UPNP - Multiple Remote Windows XP/ME/98 Vulnerabilities : http://www.eeye.com/html/Research/Advisories/AD20011220.html

[10] SNIA CIFS document : http://www.snia.org/English/Collaterals/Work_Group_Docs/NAS/CIFS/CIFS_Technical_Reference.pdf

[11] Named Pipes, Sockets and other IPC : http://www.public.asu.edu/~mujtaba/page4.html

[12] Direct Hosting of SMB Over TCP/IP (Q204279) : http://support.microsoft.com/support/kb/articles/Q204/2/79.ASP

[13] CIFS : Common Insecurities Fail Scrutiny : http://downloads.securityfocus.com/library/cifs.txt

[14] DCE 1.1 : Remote Procedure Call : http://www.opengroup.org/onlinepubs/9629399/

[15] RPC tools : http://razor.bindview.com/tools/desc/rpctools1.0-readme.html

[16] dcetest : http://www.atstake.com/research/tools/index.html#dcetest

[17] DCE/RPC over SMB : Samba and Windows NT Domain Internals. Luke Kenneth Casson Leighton. Macmillan Technical Publishing, 2000.

[18] PipeList : http://www.sysinternals.com/ntw2k/info/tips.shtml

[19] Windows 2000, Null Sessions and MSRPC. Todd Sabin, RAZOR Team, Bindview. http://razor.bindview.com/publish/presentations/nullsess.html

[20] Administration distante de systèmes Windows (Partie 2) - rpcclient : http://www.hsc.fr/ressources/breves/remote_windows_rpcclient.html