Dialogues de Log...

■ Laurent BARDI, lolo@ipbs.fr
Institut de Pharmacologie et de Biologie Structurale, Toulouse

■ Introduction

De nos jours, l'avènement du système WindowsNT dans les organismes de recherches et d'enseignement semble inéluctable. Bien qu'historiquement ces organismes soient plutôt tournés vers l'utilisation de systèmes UNIX, nous devons prendre en compte l'intégration de ce nouveau système au sein de notre communauté.

De plus, les impératifs de sécurité dus à l'utilisation extensive d'Internet nous obligent à être vigilants vis-à-vis de toutes les machines que nous possédons et qui sont visibles (entièrement ou partiellement) depuis l'extérieur.

Une des premières choses à faire dans ce cas est de vérifier (même a posteriori) ce qui se passe sur nos machines. Ceci est rendu possible par l'utilisation et l'activation des procédures de « Trace » (LOGGING). UNIX et WindowsNT disposent, chacun, d'un tel système. Mais malheureusement, ces deux systèmes ne dialoguent pas de manière spontanée. Il convient ici, de noter, que les guerres idéologiques sur le « meilleur des systèmes » (ou le moins pire...) ne seront pas abordées.

■ La vision UNIX

La notion de trace sous UNIX est gérée par le sous-système SYSLOG. Celui-ci est considéré comme un ensemble de fichiers vu de l'appelant. Afin de pouvoir centraliser ces fichiers, le système SYSLOG est constitué d'un démon (SYSLOGD) qui est en écoute permanente sur le port UDP 514 et est capable d'envoyer des informations sur ce même port. Les informations sont en fin de compte des lignes de messages (la ligne ne devant pas excéder 1024 caractères) qui possèdent le format suivant : *<XXX> MSG* (<XXX> étant l'en-tête). Les messages sont classés en fonctionnalités (FACILITIES : *KERN, USER, MAIL, DAEMON, AUTH, SYSLOG, LPR, NEWS, UUCP, CRON, AUTHPRIV, FTP, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7,…)* et en priorités (PRIORITIES : *EMERG, ALERT, CRIT, ERR, WARNING, NOTICE, INFO, DEBUG*). Le terme fonctionnalité désigne la classe du message et le terme priorité sa gravité. Les priorités sont codées sur les 3 premiers bits d 'un octet et les fonctionnalités sur les 5 bits restants (au format LSB). Cet octet est ensuite transformé en une chaîne de caractères qui constitue l'en-tête. Par exemple, un message de classe LOG_LOCAL0 (16<<3) et de gravité INFO (6) aura comme en-tête : 128+6 ⇒ *<134>*.

■ La vision WindowsNT

La notion de trace sous WindowsNT est gérée par le sous-système EventLog. Ce système est comme la plupart des sous-systèmes NT basé sur des appels RPC. Cela implique qu'une machine WindowsNT ne sait pas envoyer des informations à une autre machine (WindowsNT ou non). Par contre, la machine WindowsNT (si elle dispose des droits suffisants) est capable d'aller interroger une de ses consœurs. Cela peut se résumer par le fait de pouvoir ouvrir les fichiers de trace d'une machine WindowsNT (distante ou non), sans pour cela être capable de les mixer entre eux. De plus dans le modèle WindowsNT, il existe trois fichiers de trace par machine. Il semble possible de pouvoir en déclarer d'autres, mais la gestion de ces nouveaux fichiers ne peut être prise en compte par les outils standard de WindowsNT (EventViewer). Les trois fichiers de base correspondent à des fonctionnalités particulières: le fichier "Security" est le fichier concernant la sécurité du système dans son ensemble, le fichier "System" est dédié aux applications venant se greffer au système WindowsNT (pilotes), le fichier application est utilisable par tous mais il est réservé aux applications. Il faut cependant faire quelques remarques à ce stade :

Le fichier "Security" ne peut être entièrement manipulé que par le compte LocalSystem. Ce compte est lui-même restreint car il ne peut accéder à toutes les primitives réseau. Ce fichier est surtout utilisé pour l'"Accounting" (sessions, objets).

Le fichier "System" est surtout utilisé par les sous-systèmes de WindowsNT (pilotes) ainsi que la plupart des « services » WindowsNT. (Un « Service WindowsNT » est l'équivalent d'un démon UNIX).

Le fichier « Application » reçoit toutes les informations de fonctionnement des différents programmes.

Les comptes utilisateurs et les privilèges relatifs au service Eventlog sont donnés ci-dessous.

Compte utilisateur	Application	**Security(*)**	**System**

LocalSystem	Lire Ecrire Effacer	Lire Ecrire Effacer	Lire Ecrire Effacer
Administrateur	Lire Ecrire Effacer	Lire Effacer	Lire Ecrire Effacer
Opérateur de Serveur	Lire Ecrire Effacer		Lire Effacer
Tout le Monde	Lire Ecrire	Lire Effacer	Lire

(*) Si le compte utilisateur possède le privilège SE_AUDIT_NAME, il peut lire et effacer le fichier d'événement Sécurité.

Chaque fichier précédemment cité se compose d'événements (équivalents du « message » du SYSLOG UNIX). Chaque événement est décrit par la structure suivante :

```
Typedef struct _EVENTLOGRECORD {
DWORD Length;
DWORD Reserved;
DWORD RecordNumber;
DWORD TimeGenerated;
DWORD TimeWritten;
DWORD EventID;
WORD EventType;
WORD NumStrings;
WORD EventCategory;
WORD ReservedFlags;
DWORD ClosingRecordNumber;
DWORD StringOffset;
DWORD UserSidLength;
DWORD UserSidOffset;
DWORD DataLength;
DWORD DataOffset;
//
// Then follow:
//
// TCHAR SourceName[]
// TCHAR Computername[]
// SID UserSid
// TCHAR Strings[]
// BYTE Data[]
// CHAR Pad[]
// DWORD Length;
//
} EVENTLOGRECORD;
```

Il apparaît ici plusieurs choses relativement importantes:

- Les données ne sont pas fixes et sont définies par `StringOffset` et `DataOffset`. Ceci dans un but de pouvoir définir une structure de travail et garder la possibilité d'avoir des informations de longueur variable.
- Contrairement à UNIX, il est prévu de base de gérer le nom du compte utilisateur qui génère l'événement.
- Il y a la possibilité d'avoir des données sous forme binaire ou texte.
- Il existe 6 possibilités pour `EventType(SUCCESS, ERROR, WARNING, INFORMATION, AUDIT_SUCCESS, AUDIT_FAILURE)`.
- IL NE SUFFIT PAS DE REMPLIR CES CHAMPS POUR OBTENIR UN EVENEMENT EN BONNE ET DUE FORME !
- En fait, pour ne pas surcharger la taille des fichiers d'événement, les `Strings` ne contiennent pas toute l'information mais seulement la partie « dynamique ». Le reste est renseigné par une DLL via la base de registre.
- Le champ `EventCategory` doit être renseignée par une DLL via la base de registre.
- Le champ `SourceName` permet de récupérer correctement les deux informations précédentes en allant interroger la base de registres.
- Le champ `Computername` permet de savoir quel est la machine qui a généré l'événement.

En ce qui concerne les « traductions » (`EventCategory, Strings`) le fonctionnement est déterminé par le schéma suivant :

A chaque fois que l'on veut lire un événement, on fait appel à la fonction `ReadEventLog`.
Mais celle-ci ne retourne que des "strings". Il faut alors faire appel à la fonction `FormatMessage` pour obtenir la chaîne de caractères complète qui sera alors une combinaison du message de la DLL et de la ou des "string" lues dans l'événement.
La catégorie ne sera affichée sous format texte dans l'*EventViewer* que si catégorie.dll est existante et si le numéro de la catégorie est inférieur a `CategoryCount`.
La clef `TypesSupported` permet de définir quels sont les types de messages supportés (au maximum 7)

MESSAGE.DLL
MessageIdTypedef=DWORD

MessageId=0x1000
SymbolicName=MSG_CUSTOM
Language=English
%1
.
MessageId=0x1001
SymbolicName=MSG_CUSTOM2
Language=French
Le service %1 est démarré
.

CATEGORIE.DLL
MessageIdTypedef=WORD
MessageId=0x24
SymbolicName=NT_KERN
Language=English
UX_KERN
.
MessageId=0x1
SymbolicName=NT_USER
Language=English
UX_USER
.
MessageId=0x2
SymbolicName=NT_MAIL
Language=English
UX_MAIL
.
MessageId=0x3
SymbolicName=NT_DAEMON
Language=English
UX_DAEMON
.

…

Base de Registres

HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDAPP\CategoryCount = 0x24
HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDAPP\CategoryMessageFile = Categorie.dll
HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDAPP\EventMessageFile = Message.dll
HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDAPP\TypesSupported = 0x7

HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDSYS\CategoryCount = 0x22
HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDSYS\CategoryMessageFile = Categorie.dll
HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDSYS\EventMessageFile = Message.dll
HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDSYS\TypesSupported = 0x3

HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDSEC\CategoryCount = 0x11
HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDSEC\CategoryMessageFile = categorie.dll
HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDSEC\EventMessageFile = message.dll
HKLM\System\CurrentControlSet\Services\EventLog\Application\SYSLOGDSEC\TypesSupported = 0x7

Journal Application	Journal Système	Journal Sécurité
Event1	Event1	Event1
Event2	Event2	Event2
EventN	EventN	EventN

■ Environnement

Le programme SyslogdNT a été écrit dans le but de faire communiquer les fichiers de trace entre machines disposant d'un sous-système SYSLOGD UNIX et le sous-système EventLog de WindowsNT.

Plusieurs solutions étaient possibles, soit utiliser le fonctionnement SYSLOGD UNIX et le « porter » sur WindowsNT, soit utiliser le sous-système EventLog et le « porter » sous UNIX. Cette dernière solution est relativement complexe et lourde car il aurait fallu dans ce cas porter une grande partie de l'API WindowsNT… Nous avons donc choisi d'utiliser le fonctionnement SYSLOGD UNIX et de le porter sous WindowsNT.

Pour ce faire, nous utilisons le langage PERL et son portage sous WIN32 (ActivePerl BUILD 518) On peut ici se poser la question du choix du langage et surtout de son mode d'exécution (ActivePerl est un interpréteur). Une réponse subjective est : « Nous voulions prouver qu'avec PERL on peut tout faire (ou presque…) ».

Pour créer l'équivalent d'un démon sous WindowsNT, il faut utiliser la notion de « SERVICES ». Ces services ont plusieurs caractéristiques que nous ne détaillerons pas ici. Nous utilisons un "WRAPPER" qui permet à toute application WIN32 de fonctionner comme un service (srvany.exe). Malheureusement cette application ne permet pas de gérer les "SIGNALS". Comme on le voit les contraintes sont relativement importantes !

■ Installation

L'installateur est un programme Perl qui permet de copier les fichiers, créer la configuration dans la base de registre, créer le compte utilisateur pour le service, créer le service, le configurer et le lancer. Cette portion du programme peut paraître anodine, mais en fait c'est elle qui est la plus complexe, il faut :

Gérer les éventuels « noms longs » dans les chemins d'installation :

- Bien que « normalement » l'API WIN32 sache supporter les « noms longs » voire les UNC, cela n'est pas le cas dans l'API du sous-système EventLog (fonction `BACKUPEVENT`).
- Gérer correctement les droits sur les fichiers installés.
- Trop d'applications s'installent en copiant des DLL dans les répertoires systèmes sans même vérifier si ces mêmes DLL n'existent pas déjà; de plus la plupart des programmes soi-disant WIN32 ne font aucun appel à la pose de droits (LECTURE/MODIFICATION/TOUT) sur les fichiers installés…
- Gérer correctement les droits sur les clefs et ruches de la base de registre, les informations stockées dans la base de registre peuvent être essentielles voire critiques (c'est notre cas) il est donc impératif de protéger l'accès à celles-ci.
- Créer le compte utilisateur pour le service : Créer un compte utilisateur pour un service est un des pièges, au niveau sécurité, qui peut être fatal… Si l'on considère un réseau WindowsNT avec 3 machines (un PDC machine « A », un BDC machine « B » et un serveur machine « C », les trois machines étant dans le même domaine D), la première question à se poser est: doit-on créer un compte utilisateur local ou global? En ce qui concerne la machine « C », il est fortement conseillé d'utiliser un compte utilisateur local car si la machine est corrompue cela n'influe pas sur le domaine. En ce qui concerne les 2 DC (« A » et « B ») les comptes utilisateurs sont obligatoirement globaux (seuls les groupes peuvent êtres locaux ou globaux). La création du compte utilisateur sur un des 2 DC, « A » par exemple, implique automatiquement la création sur « B », car la base SAM est partagée par tous les DC du domaine.
- Régler les privilèges du compte utilisateur (fonctionner en tant que service, accéder aux journaux).
- Créer le service : La création d'un service fonctionnant sous un compte utilisateur nécessite la connaissance du mot de passe de ce compte utilisateur. Si l'on reprend l'exemple précédant, la création sur la machine « A » ne pose aucun problème, mais la création sur la machine « B » nécessite la connaissance du mot de passe généré à partir de la machine « A »… Nous avons donc été obligés de stocker le mot de passe du compte utilisateur dans la base de registre. Cette solution n'est pas encore satisfaisante.
- Modifier les clefs de registres relatives aux messages et catégories de messages dans "services/eventlog".
- Démarrer le service.

Il va de soi que si nous nous occupons de l'installation, nous devons aussi nous occuper de la désinstallation. Nous ne détaillerons pas les étapes car la plupart des problèmes sont les mêmes que ceux de l'installation.

SyslogdNT

L'application en elle-même fonctionne sous deux modes distincts et exclusifs.

Client

Le mode client, permet de relire, traiter, sauvegarder les fichiers d'événements déjà existants, puis ensuite de rester en attente d'événements qui seront traités aussitôt. Le processus de translation entre un événement WindowsNT et un message SYSLOGD est géré par deux tableaux associatifs (NT2UNIX_LEVEL et NT2UNIX_TYPE). Ces deux tableaux sont la charnière de l'application car ils définissent les correspondances. Chaque événement WindowsNT est traité puis envoyé via un client UDP sur le port 514 vers la machine cible. Il

est à noter que dans ce mode, aucun nettoyage des fichiers d'événement n'est effectué (les événements sont gardés en local et doivent êtres gérés par le service "EventLog"). De plus les trois fichiers d'événements sont traités (Application, Security, System).

Serveur

Le mode serveur est en fait un serveur UDP qui écoute en permanence sur le port 514. A chaque fois qu'il reçoit un message, il va traiter ce dernier en fonction de deux tableaux associatifs (UNIX2NT_LEVEL et UNIX2NT_TYPE) qui définissent le processus de translation du SYSLOGD UNIX vers les événements WindowsNT. Il est à noter ici que dans le processus de translation les messages UNIX sont tous envoyés vers le fichier d'événements « Application ». Ceci est dû à un choix de fonctionnement du service: pour pouvoir accéder aux autres fichiers, nous aurions du fonctionner avec le compte utilisateur "LocalSystem", mais ce dernier est limité en ce qui concerne les primitives réseaux. Il aurait donc fallu démarrer le service sous "LocalSystem" puis faire un « dédoublement de personnalité » pour utiliser les primitives réseaux et redevenir nous-mêmes lorsque besoin est. Hormis le coté schizophrénique de la chose, la fonction `ImpersonateLoggedOnUser` est relativement gourmande en temps et, donc, ne nous convenait pas.

Une remarque importante doit être faite ici, l'application SyslogdNT ne devrait pas être utilisée dans le cas de réseaux de machine uniquement WindowsNT, ceci pour plusieurs raisons :

- Le mode client et le mode serveur ne sont pas bijectifs, d'une machine WindowsNT « A » en mode client vers une machine WindowsNT en mode serveur « B »; les fichiers d'événements "Security" et "System" de « A » seraient redirigés vers le fichier d'événements « Application de « B » ce qui pourrait engendrer de grandes confusions…
- Notre but n'est pas d'écrire les outils indispensables qui auraient dus êtres livrés avec le système. (Prétendre avoir un système C2 dont on ne peut centraliser les fichiers de traces est assez cocasse!)

La situation à l'IPBS

Nous disposons de plusieurs serveurs WindowsNT (4 PDC, 1 BDC, 1 serveur) et de plusieurs machines UNIX (DIGITAL, SGI, LINUX). Notre politique est de tracer tout sur une machine UNIX sur laquelle nous effectuons des traitements de ces fichiers.

Nous avons donc installé SyslogdNT en mode client sur nos serveurs WindowsNT et leur avons donné comme cible la machine UNIX. Après quelques déboires (bug `syslog.auth` : WindowsNT n'était pas en cause !), le service s'est mis à fonctionner. Pour augmenter le volume des messages, nous avons mis en œuvre les audits sur le PDC et nous avons tracé toutes les connections sur notre serveur (nous avons 200 micros qui doivent obligatoirement se valider sur le domaine).

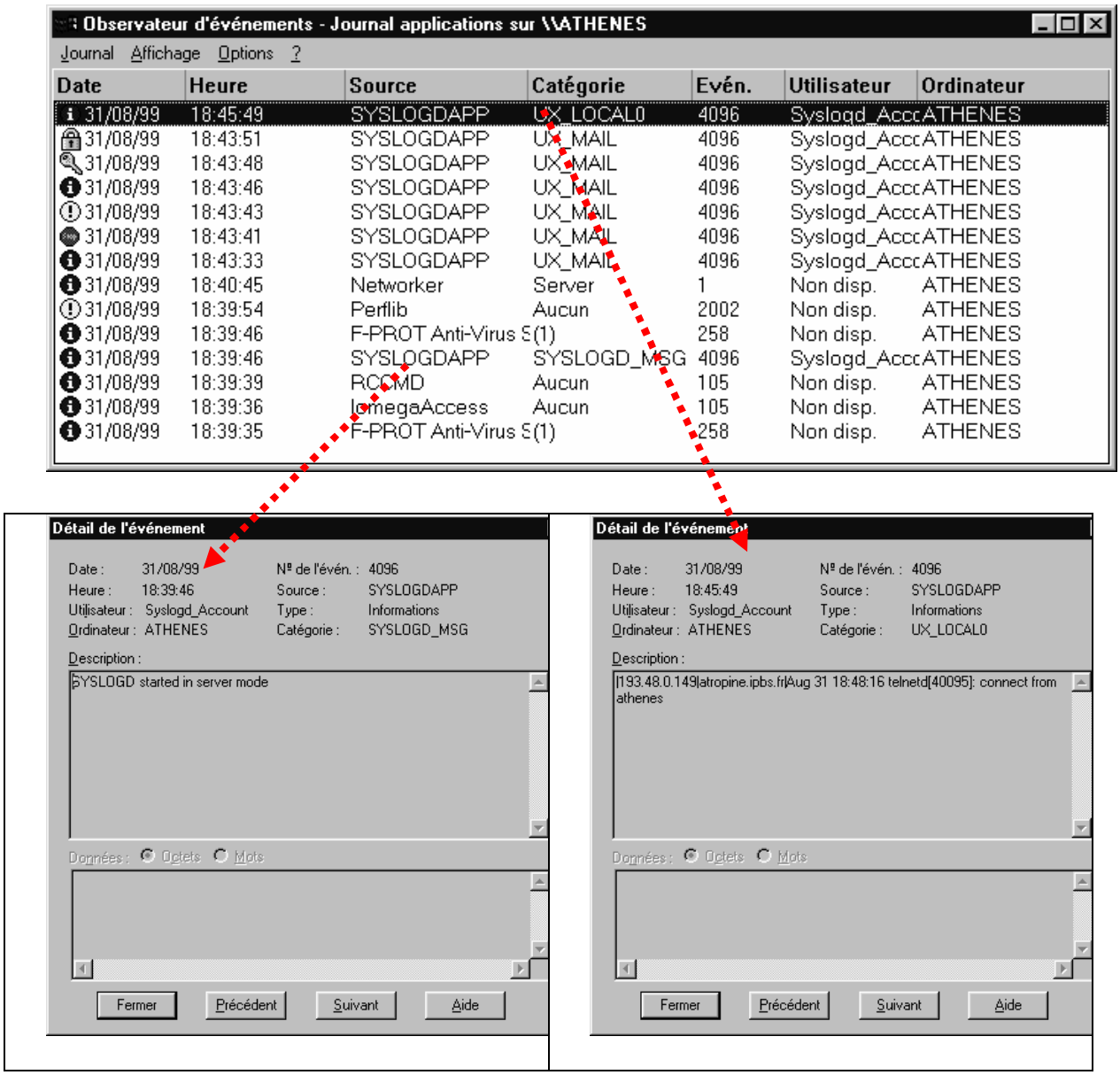
Vision UNIX

Ci-joint un exemple de trace « audit » de WindowsNT prise sur la machine UNIX :

```
Aug 31 18:47:14 paris Security[528]: Sessions acceptées : Nom de l'utilisateur : XXXXX Domaine : IPBS N: de la session : (0x0,0x102F547F) Type de session : 3 Processus d'ouverture de session : KSecDD Lot d'authentification : MICROSOFT_AUTHENTICATION_PACKAGE_V1_0 Nom de station de travail : \\YYYYYY

Aug 31 18:47:14 paris Security[538]: Fermeture de la session utilisateur : Nom de l'utilisateur : ZZZZZZ Domaine : IPBS N: de la session : (0x0,0x102F4D3A) Type de session : 3

Aug 31 18:49:15 paris Security[538]: Fermeture de la session utilisateur : Nom de l'utilisateur : TTTTTT Domaine : IPBS N: de la session : (0x0,0x102F547F) Type de session : 3

Aug 31 18:49:16 paris Security[528]: Sessions acceptées : Nom de l'utilisateur : UUUUUU Domaine : IPBS N: de la session : (0x0,0x10301565) Type de session : 3 Processus d'ouverture de session : KSecDD Lot d'authentification : MICROSOFT_AUTHENTICATION_PACKAGE_V1_0 Nom de station de travail : \\VVVVVVVV

Aug 31 18:49:16 paris Security[538]: Fermeture de la session utilisateur : Nom de l'utilisateur : WWWWWW Domaine : IPBS N: de la session : (0x0,0xFEF03B4) Type de session : 3
```

Vision WindowsNT

Nous avons aussi installé SyslogdNT en mode serveur sur une machine WindowsNT et lui avons envoyé une partie des traces de notre routeur et de certaines machines UNIX. Là aussi, les résultats, bien que peu exploitables par manque d'outils natifs ont été satisfaisants. Ci joint des copies d'écrans :



Conclusions, le FUTUR ?

Au jour d'aujourd'hui, cela fonctionne de façon satisfaisante, l'installation sur une nouvelle machine se fait rapidement et même si WindowsNT ne nous épargne pas ses jolis écrans bleus, le service redémarre automatiquement.

Il est évident que cette application peut évoluer (et nous le souhaitons), on peut citer entre autres :

- Mode Client & Serveur
- Cryptage des données
- Vérification de la source émettrice
- Configuration avec mode GUI
- Visualisation avec pose de filtres en mode ligne de commande et GUI
- …

■ Références

- Le programmeur PERL5, SIMON ET SHUSTER, MACMILLAN (http://www.ssm.fr/ISBN 2-7440-0172-x).
- Jean-Marie Rifflet ; La communication sous UNIX: Applications réparties, 2e édition ; Ediscience International, 1995.

- http://www.activestate.com
- http://msdn.microsoft.com/library/sdkdoc/winbase/eventlog_2tbb.htm

http://msdn.microsoft.com/library/periodic/period98/html/thenteventlogpart1meetapi.htm
http://www.scit.wlv.ac.uk/~cm1924/scitcd/modules/cp3010/lectures/services/html/lectserv.html

- http://search.microsoft.com/us/dev/default.asp
- http://www.roth.net/perl/
- http://jenda.krynicky.cz/#Win32::Lanman
- http://www.zurich.ibm.com/~deb/Enst_V102/
- http://theory.uwinnipeg.ca/CPAN/data/Net-Daemon/Net/Daemon/Log.html

ftp://ftp.ipbs.fr