Securite sous Windows VISTA. Problematique et impact sur le developpement d'applications.

Mémoire

Windows VISTA apporte des changements significatifs en termes de sécurité. Ces évolutions causent plusieurs difficultés de gestion en entreprise, ainsi que pour le développement d'applications.

Emmanuel Chiarello

13/12/2008

Convention typographique utilisées dans ce document

Convention	Utilisée pour
TITRE 1	Titre des chapitres principaux
Titre 2	Chapitres de niveau 2
Titre 3	Chapitres de niveau 3
Titre 4	Chapitres de niveau 4
Titre 5	Chapitres de niveau 5
Sous-titre	Sous-titres
Acronymes	Abréviations usuelles, référencées dans l'index final.
Méthodes et API	Nom de méthodes et autres mots lies aux aspects de programmation.
Important	Résumé intermédiaire présentant les aspects importants développés dans le chapitre.
Sous-titre	Sous-titre présentant un aspect particulier et/ou répétitif.
Extrait de texte	Texte provenant d'une source externe et repris plus ou moins tel quel.
Référence	Référence, à un chapitre, une illustration ou un tableau du document.
Accentuation	Mot ou ensemble de mots ayant une importance particulière.
« Locutions »	Ensemble de mots non divisibles devant être pris en tant que forme globale. Peuvent ne pas être traduits.

CAHIER DES CHARGES DETAILLE

Contexte

SIG (Services Industriels de Genève) est une entreprise genevoise autonome de services publics, active dans la fourniture de l'eau, électricité, gaz, chauffage à distance, traitement des déchets et télécommunications.

Le parc informatique de SIG est composé d'environ 2000 PC et 150 serveurs. La télédistribution des applications sur les postes de travail et les serveurs Windows se fait à l'aide d'un programme baptisé « SIGTL » développé à SIG à partir de 1997 pour Windows 3.11 et NT4 puis Windows 2000 et XP. Le remplacement de cet outil a été envisagé en vue du passage à Windows VISTA. Toutefois, compte tenu de la charge et des délais engendrés par le remplacement d'un tel outil, il semble économique et préférable pour l'heure d'adapter l'outil SIG.

D'autant plus que les connaissances acquises durant ce travail seront de toutes évidences un atout dans le cadre du déploiement de Windows VISTA.

Outil de télédistribution

L'intérêt d'un outil de télédistribution est d'optimiser le déploiement des applications, principalement sur les postes de travail mais aussi sur les serveurs. Les applications sont installées automatiquement (idéalement sans aucune intervention humaine) et gérées de manière centralisée. La télédistribution des applications s'inscrit dans un processus administratif de « demande d'accès aux applications » où les applications nécessaires aux utilisateurs et après validation, sont poussées vers l'ordinateur concerné. Dans un mode self-service, l'utilisateur accède à un catalogue d'applications optionnelles (en accès libre), qu'il peut choisir d'installer suivant ses besoins.

D'autres tâches de maintenances des postes de travail peuvent nécessiter les services de l'outil de télédistribution comme l'adaptation des paramétrages du firewall personnel, l'installation de « service pack », etc.)

Les principales fonctionnalités sont :

- « Packaging » des applications dans des unités de distribution unitaires (les paquets)
- Définition des interdépendances entre paquets logiciels.
- Installations conditionnées en fonction des profiles utilisateur ou de caractéristiques physiques/logicielles des ordinateurs concernés.
- Inventaire et suivi des licences.

L'outil SIGTL se compose d'une base de données Microsoft SQL Server (contenant la définition des paquets logiciels, les profiles utilisateurs et autres informations de gestion), d'une librairie de logiciels et d'un agent, installé sur les postes de travail en même temps que le système d'exploitation de base.

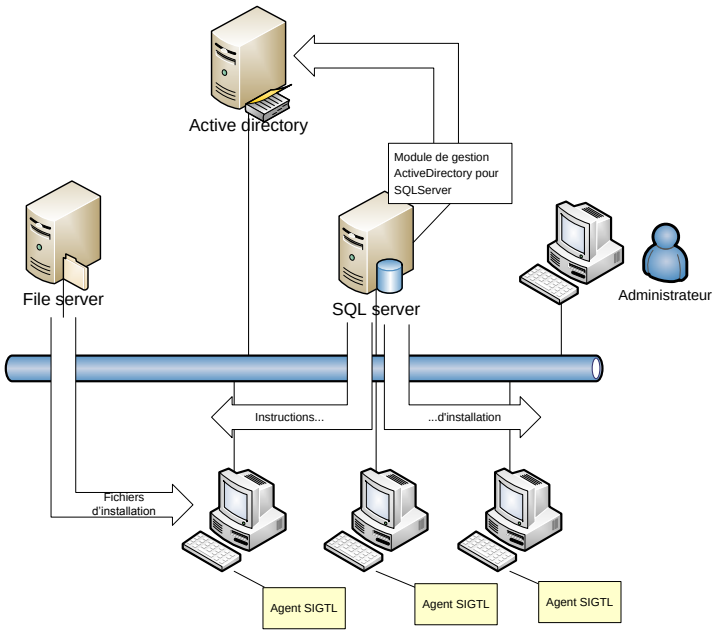


Schéma physique de l'infrastructure de télédistribution SIGTL

Objectif du travail

Cette étude a pour objectif d'étudier les mécanismes de sécurité de Windows VISTA afin de résoudre les dysfonctionnements de l'agent de télédistribution SIGTL.

En première analyse, il apparaît que les principaux blocages du programme sous Windows VISTA, se situent à deux niveaux :

1. Mécanisme de communication IPC.
   Le programme SIGTL utilise COM/DCOM pour la communication entre le processus maître et le service d'installation. Windows VISTA bloque cette communication.
2. Mécanisme d'élévation de privilèges. Le programme de télédistribution doit effectuer des tâches de niveau « administrateur », tout en accédant au contexte de l'utilisateur interactif.
   C'est précisément ce que Microsoft cherche à contrer dans Windows VISTA avec de nouveaux mécanismes de sécurité.

Les problèmes et solutions pour le déploiement des applications seront également étudiés :

- Virtualisation des applications
- Contrôle des privilèges nécessaires à l'exécution des programmes.

Application SIGTL

Cette application est développée en Pascal objet, dans l'IDE Delphi de Borland.
SIG utilise actuellement la version Turbo Delphi 2006, qui souffre malheureusement de l'absence de support de Windows VISTA. Il n'est toutefois pas envisagé d'acquérir Delphi 2007 pour l'instant, considérant que les nouvelles fonctionnalités ne seront pas pertinentes dans le cadre de ce projet.

HISTORIQUE

Depuis 1997, différentes évolutions ont été nécessaires pour adapter le programme SIGTL aux évolutions de Windows :

1997 Support Windows 3.11 (16 bits) et Windows NT4 (32 bits). Développement initial sous Delphi, code mixte compilable 16 et 32 bits.

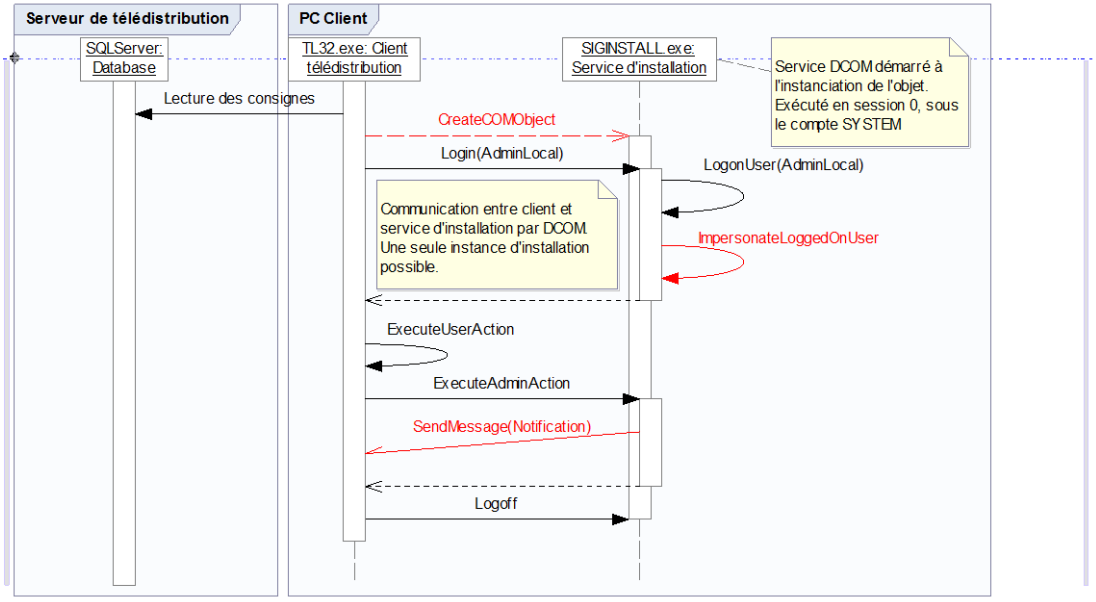
2001 Support Windows 2000 et installation sans droit administrateur. Développement d'un service COM pour l'élévation des privilèges.

2004 Support Windows XP. Adaptation du service COM.

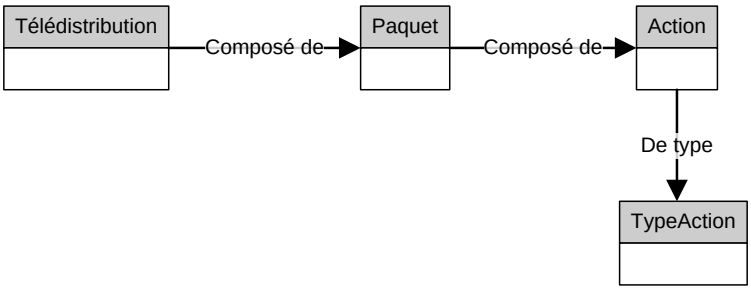
2008 Réflexion pour le support de Windows VISTA.

ARCHITECTURE

Le diagramme d'activité suivant indique les interactions entre le composant en mode utilisateur (client de télédistribution) et le service d'installation COM exécuté sous le compte SYSTEM (en rouge, les appels en échec ou potentiellement en échec sous Windows Vista)



Vue synthétique des données, une télédistribution peut être représentée avec les entités et relations suivantes :



L'action représente l'unité d'exécution la plus fine.
Les principaux types d'actions utilisables sont les suivants :

EXEC	Exécution d'un programme, en mode console on récupère la sortie standard dans le journal d'exécution de la télédistribution. L'exécution du programme peut être synchrone ou asynchrone au choix
ICON	Permet la gestion des raccourcis de l'explorateur Windows
COPY	Permet l'installation de fichiers au sens général. Copie du fichier, contrôle de version, contrôle des fichiers protégés par le système, prise en charge des fichiers ouverts, extraction de fichiers compressés par compress.exe de Windows.
REG	Mise à jour des clés et valeurs des clés du registre Windows (HKCU, HKCR et HKLM). La ruche HKCU est mise à jour dans le contexte de l'utilisateur exécutant l'installation ainsi que dans l'ensemble des profiles du PC. Les nouveaux profiles héritent des changements reportés dans HKU\.DEFAULT
MSI	Pilotage de l'installation, configuration, désinstallation d'un package au standard Windows Installer.
…	

D'autres actions complètent la panoplie d'actions utilisables pour le « packaging » d'applications.

Accessoirement, certaines actions permettent le déploiement d'applications sur Pocket PC (WinCE 2003- WinCE 6) : Copie de fichiers, Modification de registre, gestion des raccourcis, installation de package au format Windows CE (.CAB). Toutefois, les contraintes de sécurité de la plateforme Windows CE sont très différentes de celles liées aux versions Windows des postes de travail, sujet de ce travail.

PROBLEMES SOUS WINDOWS VISTA

Le premier problème rencontré se situe au niveau de la communication COM entre le programme principal (TL32.EXE) et son service d'installation (SIGINSTALLER.EXE)
Il est probable que le problème soit dû à l'isolation des services Windows dans une session 0, différente de celle de l'utilisateur et non interactive. Il en résulte une attente bloquante (plusieurs minutes en tout cas) de la part du programme client envers le service d'installation COM.

Le second problème se situe au niveau des actions de télédistribution devant exécuter un programme en mode privilégié et interactif.
Lorsque le programme principal (TL32.EXE) exécute le serveur COM en tant que processus standard (non service), il est possible d'effectuer une élévation de droits du processus (LogonUser() et ImpersonateLoggedOnUser()) sous VISTA. Toutefois, le processus se retrouve avec un jeton limité et un niveau d'intégrité « Medium », ne permettant pas l'exécution correcte d'un programme d'installation.

A noter que le processus de télédistribution doit pouvoir se dérouler avec un minimum d'interaction utilisateur (idéalement aucune). Une invite d'élévation telle que présentée par Windows VISTA n'est pas acceptable.

Analyse

ELEVATION DE PRIVILEGE

Le programme de télédistribution doit être utilisable sur les plateformes Windows 2000, XP/2003 et VISTA/2008. Il n'est pas envisagé de construire des versions différentes du programme SIGTL pour les différentes versions d'OS.

Remarque : Sans autre indication particulière, les informations se réfèrent au site *Microsoft MSDN*

Les principales fonctions de l'API Windows pour gérer le changement d'identité sont :

- LogonUser()
  Permet de créer un jeton servant à identifier et contrôler les accès d'un processus, propre à un compte utilisateur donné.
- ImpersonateLoggedOnUser()
  Permet d'emprunter, temporairement ou non, l'identité d'un jeton d'accès donné, en l'associant au thread courant.
- CreateProcessAsUser()
  Permet de créer un processus fils avec l'identité d'un autre compte utilisateur, à l'aide d'un jeton d'accès créé avec LogonUser()
- RevertToSelf
  Permet au thread courant de retrouver son identité initiale (celle du processus)

En fonction des différentes versions de Windows successives, les exigences d'exécution de ces méthodes ont évolué :

Windows 2000 :

La fonction LogonUser() nécessite le privilège Windows très spécial SE_TCB_NAMES. Ce privilège traduit comme « Agir en tant que partie du système d'exploitation » est le privilège ultime sous Windows et n'est disponible par défaut que pour les comptes membres du groupe Administrateurs ainsi que pour le compte SYSTEM, utilisé pour le démarrage des Services Windows.

La conséquence de cette contrainte fait qu'une élévation de privilège ne peut être faite que par l'intermédiaire d'un programme auxiliaire fonctionnant comme service et démarrant avec le compte SYSTEM (plus aisé à gérer qu'un compte administrateur spécifique car ne nécessitant pas de mot de passe connu)

Windows XP :

Windows XP ne requiert plus le privilège SE_TCB_NAMES pour LogonUser(). Ainsi, un processus utilisateur peut changer par lui-même d'identité, sans recourir à un processus/service tiers.

Windows VISTA :

Avec l'introduction de l'UAC (Contrôle de compte utilisateur), Windows VISTA ajoute de nouvelles barrières pour contrer l'élévation de privilèges. La partie la plus visible d'UAC est l'invite de validation des actions nécessitant des droits élevés.

Toutefois, de nombreux mécanismes liés à la sécurité sous Vista posent problème :

- Jetons d'accès limités pour les comptes administrateurs
- Niveaux d'intégrité, empêchant strictement à un processus de plus faible niveau d'enfanter un processus de plus haut niveau.
- Isolation des services dans une session séparée.
- Impossibilité d'élever les privilèges en ligne de commande (commande runas.exe)

Ces notions sont brièvement expliquées ci-dessous et nécessiteront une analyse approfondie lors du travail final.

OBJETS DE SECURITE WINDOWS

Ce chapitre liste et décrit brièvement les objets de sécurité Windows à approfondir dans le cadre de ce projet.

Jeton utilisateur

Au cœur de la sécurité Windows, se trouve une structure de données appelée « Access Token ». Cette structure, accessible uniquement par l'intermédiaire d'un handle contient les informations d'identité, de privilèges et de droits d'accès liés à un utilisateur ayant ouvert une session.

Chaque processus exécuté sous Windows est associé à un « Access Token », permettant le contrôle d'accès aux ressources protégées par des ACL (Access Control List) ou aux fonctions systèmes contrôlées par les privilèges.

Un thread peut également posséder son propre jeton, dans le cas d'un emprunt d'identité (appel d'une des méthode Impersonate...() de l'API Windows)

Le jeton utilisateur étant au cœur de la sécurité Windows, une bonne maîtrise des fonctions de manipulation du « Token » est nécessaire.

ACL

Liste de contrôle d'accès. Structure de données sous forme de liste, attachée à tout objet Windows « sécurisable ».

Il sera certainement nécessaire, dans le cadre de ce projet, de contrôler les ACL d'objets USER, comme « Windows station » et « Desktop ».

Privilèges

Sortes d'étiquettes prédéfinies, permettant d'affecter, de manière particulièrement souple, les « privilèges systèmes » tels que « arrêter l'ordinateur », non sécurisable par ACL. Les privilèges sont affectés par défaut à certaines catégories d'utilisateurs, mais un administrateur peut, de manière locale ou centralisée, modifier l'attribution des privilèges. Certains privilèges requièrent la plus grande prudence dans la modification de leur affectation, comme p.ex. SE_TCB_NAMES (Agir en tant que partie du système d'exploitation). Certains privilèges attribués sont désactivés par défaut. Pour être effectif, le programme ayant besoin de ce privilège devra au préalable l'activer (AdjustTokenPrivileges())

Les privilèges requis par certaines fonctions Windows conditionnent fortement l'architecture de l'application.

SID

Un utilisateur ou un groupe est représenté dans Windows par un « SID ». Un SID est une structure de données qui, représentée textuellement ressemble à ceci :

S-1-5-18
S-1-5-21-3043594363-1568931741-1397882902-1007

Un SID est composé d'une suite variable d'identifiants. Il identifie de manière unique, sur un ordinateur donné ou sur un domaine donné, un utilisateur ou un groupe.

La structure du SID est un héritage des débuts de Windows NT.

Certains SID « biens connus » sont identiques sur chaque ordinateur. Les autres, appartiennent à l'ordinateur ou au domaine sur lequel ils ont été générés.

Un fichier protégé par une ACL (Liste de contrôle d'accès) faisant référence à un SID d'un ordinateur ou domaine étranger, n'aura pas de sens (SID inconnu)

Les SID interviennent dans de nombreuses API Windows. Les « niveaux d'intégrité » sous Windows VISTA se réfèrent à des SID « biens connus ».

Niveau d'emprunt d'identité

« Impersonation level »
Le niveau d'emprunt d'identité attaché au jeton d'accès détermine l'usage qui pourra être fait de ce jeton :

SecurityAnonymous	Le serveur ne peut ni authentifier ni emprunter l'identité du client.
SecurityIdentification	Le serveur peut connaître l'identité et les privilèges du client, mais ne peut pas en emprunter l'identité.
SecurityImpersonation	Le serveur peut emprunter l'identité (le contexte de sécurité) du client, sur le système local.
SecurityDelegation	Le serveur peut emprunter l'identité et le contexte de sécurité du client, sur un système distant.

Ce point figure à titre informatif, sans certitude quand à son utilité dans le cadre du projet.

Windows VISTA

L'objectif de ce travail nécessite une pré-étude des mécanismes de sécurité de Windows Vista, afin de déterminer les grandes lignes du déroulement du projet.

Ce chapitre recense les principaux mécanismes mis en place dans Windows VISTA. Pour chacun, une évaluation de sa pertinence pour ce travail est mentionnée.

SAL

Suite aux expériences désastreuses de Windows XP en matière de sécurité, ayant débouchées sur une panoplie de « rustines » fournies dans le SP2, le développement de Windows VISTA a été fortement orienté sur l'axe sécurité. Jamais Microsoft n'avait lancé d'opération de cette envergure sur le sujet de la sécurité. Il en allait probablement pour Microsoft, de sa survie face à des systèmes concurrents comme Linux, certes moins exposée en termes de masse, mais certainement mieux conçus.

L'initiative Microsoft pour la sécurité des applications part du compilateur et de l'intégration de toute une batterie d'annotations pour C/C++, permettant la détection à la compilation d'une grande partie des erreurs de programmation pouvant déboucher sur une faille exploitable sous forme de buffer/stack overflow. [2]
Ce système d'annotations se nomme SAL (Standard Annotation Language)

Les annotations SAL classiques comme __in et __out, __in_opt (optionnel) et __inout, indiquent le sens d'utilisation des paramètres d'une fonction.

D'autres annotations plus subtiles rendent assez difficile l'interprétation de l'écriture des fonctions :

__inout_bcount_full(n)

Exemple :

*Void ConverToLittleEnian(__inout_bcount_full(cbInteger) BYTE *pbInteger, DWORD cbInteger, __out_opt EXCEPTION *pException) ;*

Permet de contrôler la taille du buffer *pbInteger, en fonction de la taille connue dans cbInteger

__inout_bcount_part(n, m)

Variante de __inout_bcount_full(n) permettant de définir la taille du buffer et le nombre d'octets utilisés. (__inout_ecount_part(n, m) se réfère à des éléments plutôt qu'à des « bytes »)

__deref_out_bcount(n)

L'argument de fonction marqué avec __deref_out_bcount(n) sera défini comme un buffer non initialisé de n-bytes une fois déréférencé.

SAL est accessoire pour ce projet. Une compréhension basique de SAL est toutefois nécessaire pour la traduction de certaines déclarations de fonction des API C Windows en Pascal.

DEP (Data Execution Prevention)

Ce mécanisme permet au système d'exploitation d'empêcher certaines attaques de type buffer-overflow, où un programme malveillant réussi à détourner le pointeur d'instruction de la zone mémoire de code, vers une zone de données.

Ce mécanisme est efficacement soutenu par tous les processeurs récents, et ce depuis quelques années déjà.

DEP a été introduit dans Windows XP SP2, mais il n'est toujours pas généralisé à l'ensemble des programmes fonctionnant sous Windows. Seuls les composants Windows sont ainsi protégés. L'idée étant d'empêcher les exploitations de failles du système, il semble normal de privilégier la protection du système, d'autant que certaines applications (ou compilateurs) utilisent volontairement des zones mémoire de données pour du code exécutable (par exemple génération dynamique de code machine ou compilation « just-in-time » de p-code)

A priori, DEP n'a aucun impact sur ce projet.

Type de jeton

Un jeton utilisateur sous Windows peut être de 3 types : « Default », « Limited », « Full ». Les types « Limited » et « Full » concernent les jetons liés à des comptes privilégiés (groupe administrateur ou « utilisateurs avec pouvoir ») pour lesquels 2 jetons sont créés à l'ouverture de session. Les jetons des utilisateurs non privilégiés ne sont pas dupliqués et sont de type « Default ».

Cette notion devra être étudiée en détail compte tenu de son importance dans le cadre d'une recherche d'élévation de privilèges.

NIVEAU D'INTEGRITE

Le 2ème concept de sécurité important introduit par Windows VISTA est le « niveau d'intégrité ». Un « Niveau d'intégrité » est associé au jeton utilisateur à l'ouverture de la session. Un utilisateur non privilégié reçoit un jeton de niveau « Medium » alors qu'un utilisateur privilégié (Administrateur ou Power user) reçoit un niveau « Elevé » dans son jeton complet.

Le « Niveau d'intégrité » peut être défini au niveau d'une ACL « Système », utilisée jusque-là pour la définition des audits d'accès aux objets. Une SACL (Liste de contrôle d'accès système) contenant une ACE (Entrée de contrôle d'accès) pour un niveau d'intégrité donné peut refuser la lecture, l'écriture ou l'exécution aux processus dont le niveau est égal ou inférieur au niveau spécifié et ceci, indépendamment des privilèges de l'utilisateur.

Certains dossiers systèmes, sont protégés par un SACL spécial empêchant la modification du contenu par tous processus ayant un niveau d'intégrité inférieur. Un programme de télédistribution doit pouvoir contrôler les dossiers systèmes.

Sommaire

RESUME

Ce travail se compose de 2 phases distinctes ; l'étude des mécanismes de sécurité Windows Vista et le développement du service d'installation.

Globalement, les principales réalisations sont:

- Construction d'un service d'installation avec élévation de privilège, contournant la demande de consentement et l'isolation des services en session 0 de Vista.
- Construction d'un framework de communication RPC.
- Construction d'un outil d'exploration des attributs de sécurité des objets Windows.
- Intégration du service d'installation au programme de télédistribution SIG.
- Construction de divers programmes permettant de tester les mécanismes de sécurité comme UIPI (isolation des privilèges appliquée aux objets USER) et DEP (protection contre l'exécution des zones mémoire de données).
- Extension du framework Delphi de tests unitaires, pour détecter les fuites mémoire/ handles des classes testées.

Et les technologies utilisées :

- XML pour la sérialisation des appels RPC
- Programmation concurrente (threads, et objets de synchronisation)
- Chiffrement par les « CryptoAPI » de Windows
- Sécurité et principe de moindre privilège appliqué aux services.

Ce travail démontre la faisabilité d'un mécanisme contournant le consentement utilisateur de Windows Vista. Les adaptation apportées ont également permis un gain global de qualité, en diminuant la complexité structurelle du programme existant. L'agent de télédistribution mis au point permet l'installation par télédistribution dans les environnements Windows 2000, XP et Vista.

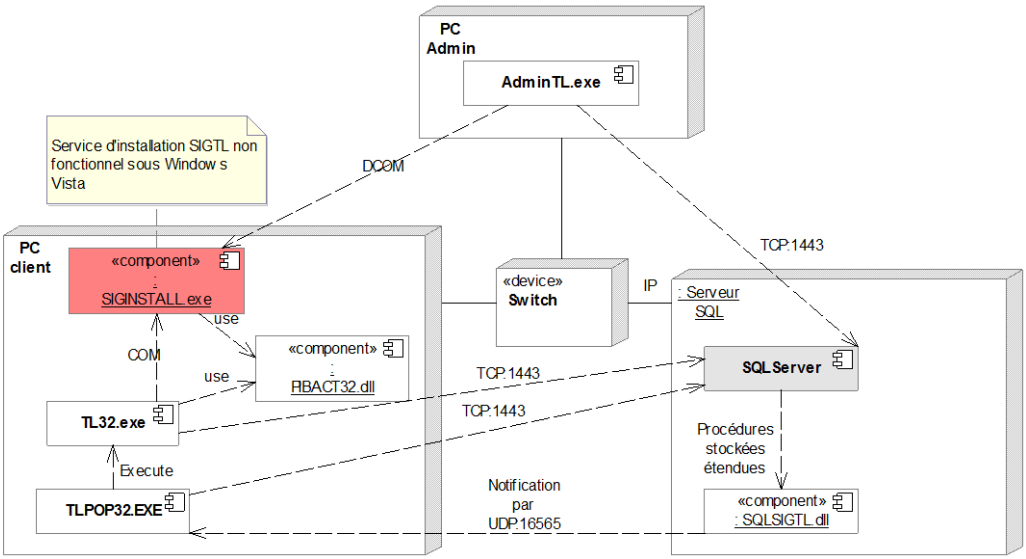


Figure 1 : Diagramme de déploiement des composants « SIGTL »

ETUDE ET DEVELOPPEMENT

Introduction

L'adaptation du programme de télédistribution « SIGTL » pour Windows Vista est un challenge important, du fait de sa forte dépendance vis-à-vis des nouveaux mécanismes de sécurité. Ce travail a donc logiquement commencé par une étude approfondie de ces mécanismes comme le contrôle de compte utilisateur (UAC) et la réalisation d'un outil d'exploration des attributs de sécurité des objets Windows (SecurityExplorer).
Nous verrons ensuite le principe d'élévation de privilèges, développé pour le service d'installation et permettant le contournement de certains mécanismes de sécurité de Vista.

Etude des mécanismes de sécurité Windows

MODELES ET CONCEPTS

Les principales responsabilités d'un système d'exploitation en termes de sécurité sont d'assurer la confidentialité et l'intégrité de l'information.

- **Confidentialité**: prévention de la divulgation non autorisée de l'information
- **Intégrité**: prévention de la modification non autorisée de l'information

Ces concepts reposent sur des modèles formels, établis dans les années 70. Aucun de ces modèles ne répond à lui seul aux besoins de sécurité. Ils sont le plus souvent utilisés en association (1).

Modèle de confidentialité

Dans le monde commercial, la confidentialité est habituellement assurée par des mécanismes moins rigoureux que ceux utilisés dans le monde militaire / sécurité nationale. Par exemple l'information peut être attribuée à un propriétaire / gardien qui en contrôle l'accès.
Le modèle *« Bell-Lapadula »* (2) est un modèle garantissant le respect des règles de confidentialité de l'information. En résumé, il défini les propriétés de lecture et d'écriture par rapport aux niveaux de confidentialité inférieur et supérieur : « No read up » (propriété de lecture simple) et « No write down » (propriété de sécurité ☆)

Modèle d'intégrité

Le modèle *« Biba »* (3) est l'inverse du modèle Bell-Lapadula, mais vise à garantir l'intégrité de l'information (ou du système). Il défini 3 règles « no read down » (axiome d'intégrité simple) , « no write up » (axiome d'intégrité ☆) et « no invoke up » (axiome d'intégrité d'invocation) Il est le plus souvent implémenté sous forme de contrôle d'accès obligatoire (par opposition au contrôle d'accès discrétionnaire)

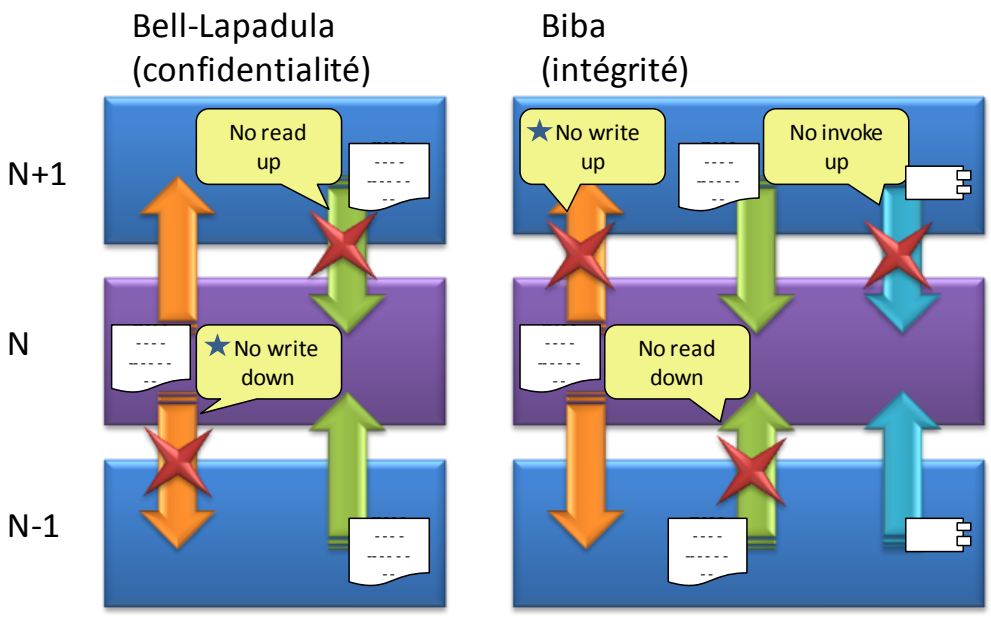


Figure 2 Règles des modèles Biba et Bell-Lapadula

Standards de sécurité

Le principal standard permettant d'évaluer le niveau de sécurité d'un système informatique est le TCSEC (Trusted Computer System Evaluation Criteria).
CC (Common Criteria), standard ISO 15408, remplace TCSEC et ITSEC (son équivalent européen) depuis 1999 en proposant un système d'évaluation basé sur le concept de « Protection Profile » (PP), à la fois plus large et plus souple que TCSEC.

A1	Design vérifié
B3	Domaines de sécurité
B2	Protection structurée
B1	Protection de sécurité labellisée (MAC)
C2	Protection d'accès contrôlée
C1	Protection d'accès discrétionnaire (DAC)
D	Protection minimale

Tableau 1 Niveaux TCSEC et leurs principales exigences

Evaluation Assurance Level	
EAL1	testé fonctionnellement
EAL2	testé structurellement
EAL3	testé et vérifié méthodiquement
EAL4	conçu, testé et vérifié méthodiquement
EAL5	conçu et testé de façon semi-formelle
EAL6	vérifié, conçu et testé de façon semi-formelle
EAL7	vérifié, conçu et testé de façon formelle

Tableau 2 Niveaux CC et leurs principales exigences

L'intérêt du TCSEC est de définir simplement les exigences fonctionnelles en matière de sécurité des systèmes d'exploitation (ou plus généralement du TCB, Trusted computing base). TCSEC défini les niveaux d'exigences croissantes de D à A.
En matière de contrôle d'accès, le niveau C exige la protection des objets par un contrôle d'accès discrétionnaire (DAC) et le niveau B, un contrôle d'accès obligatoire (MAC), en plus du DAC.

A noter que les premières exigences en termes de résistance à l'intrusion ne sont introduites qu'à partir du niveau B2 (4).

Le contrôle de confidentialité est principalement assuré par des DAC alors que le contrôle d'intégrité se fait à l'aide de MAC.

Ainsi, le niveau C du TCSEC défini les exigences en termes de confidentialité et le niveau B en termes d'intégrité.

DAC

Pour « Discretionary access control » ou contrôle d'accès discrétionnaire, est un moyen de limiter l'accès aux objets, basé sur l'identité des sujets ou des groupes auxquels ils appartiennent (5). Le DAC suppose souvent que chaque objet ait un propriétaire qui contrôle les permissions d'accès.

MAC

Pour « Mandatory access control » ou contrôle d'accès obligatoire, est une méthode de gestion des droits utilisateurs, utilisée lorsque la politique de sécurité impose que les décisions de protection des objets ne soient pas être prises par le propriétaire de l'objet. (6)

TCB

Le TCB (Trusted Computing base) se défini comme l'ensemble des composants matériels, microcode et logiciels critiques pour la sécurité du système informatique, dans le sens ou un dysfonctionnement de l'un des composants du TCB compromet la sécurité de l'ensemble du système (7). Pour cette raison, le TCB devrait idéalement être aussi réduit que possible (micro-kernel) et ses frontières en tout cas clairement définies (8). Cette définition de bon sens reste conceptuelle, mais on comprend qu'au cœur du TCB est le mécanisme de contrôle d'accès, appelé « Security Reference Monitor » ou « Reference Validation Mechanism », chargé de vérifier le flux d'information entre objets et sujets.

> Un sujet est identifié dans le système (p.ex. par un jeton d'accès).
> Un objet est protégé contre les violations d'accès (p.ex. par une liste de contrôles d'accès).

Dans l'article « Trusted System Concepts », Marshall D. Abrams généralise le paradigme du TCB suite aux changements intervenus dans les infrastructures informatiques des années 1990. Le TCB inclus d'autres fonctionnalités ne prenant pas de décision de contrôle d'accès, tel que les mécanismes d'audit, placés dans la frontière du TCB (9 p. 5).

L'introduction récente du module de plateforme sécurisée (TPM pour Trusted Plateform Module). Cette puce électronique intégré à la carte mère des ordinateurs, intègre des fonctions de chiffrement et permet d'étendre le TCB aux données binaires du système d'exploitation. Le chargement du système d'exploitation à partir d'un volume chiffré par une clé TPM, nécessite un accès privilégié aux clés de chiffrement. Cette procédure reste sûre car exécutée au sein du TCB. TPM valide les composants clé pour le démarrage (BIOS, MBR, etc.) avant de fournir une clé pour le déchiffrement des données. TPM protège ainsi les données jusque là exposées à des attaques « offline », mais n'est toutefois pas à l'abri d'attaques utilisant un accès direct à la mémoire (DMA) au niveau du bus PCI par exemple (10).

ELEMENTS DE SECURITE

Ce chapitre décrit les éléments et concept de sécurité de l'environnement Windows, dont les relations sont synthétisées dans le diagramme ci-dessous. Ces éléments sont importants et seront souvent cités plus loin dans ce document.

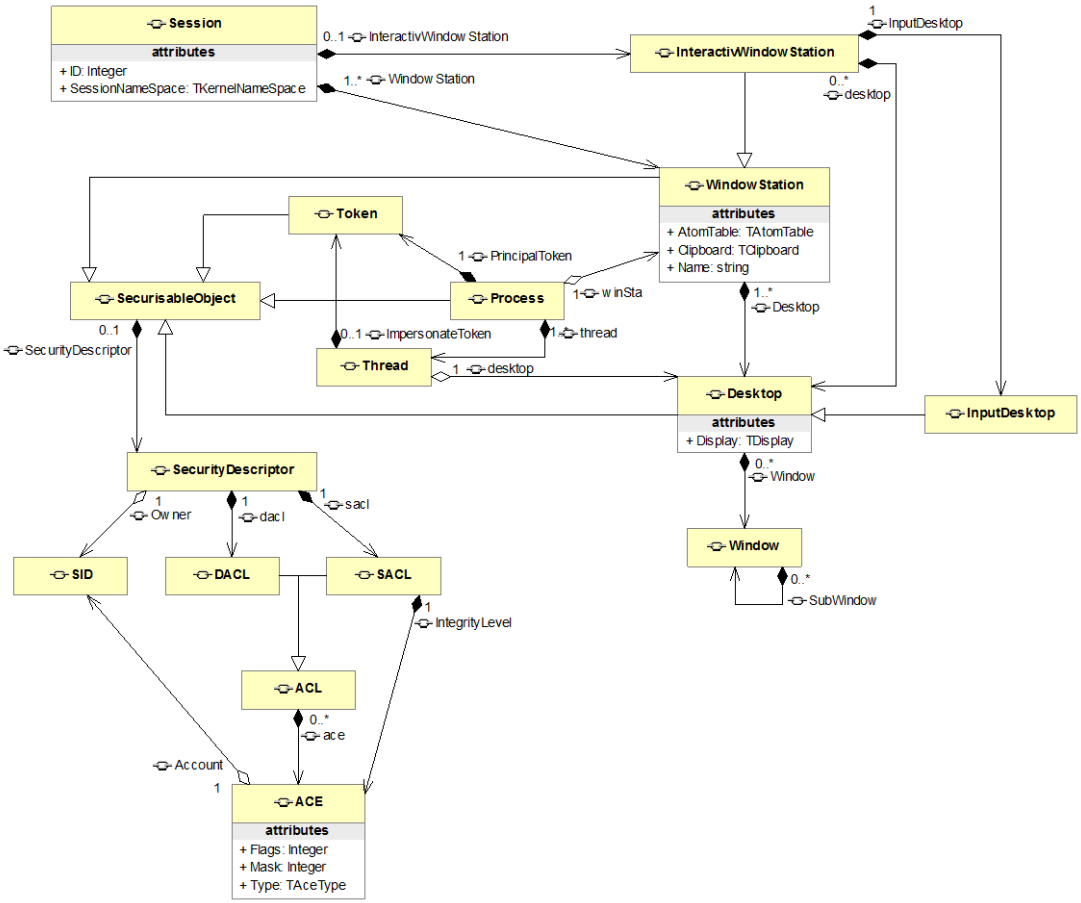


Figure 3 : Résumé des relations entre objets Windows

Identification (SID)

Un sujet (utilisateur) de même qu'un groupe de sécurité est identifié dans Windows par un SID (Security Identifier). Un SID est une structure de taille variable identifiant de manière unique dans le monde chaque sujet. Un SID est constitué des identifiants de:

- L'autorité ayant produit le SID
- Un nombre variable de sous-autorités de confiance (RID) relatives à l'autorité ayant produit le SID
- Le RID final identifiant le sujet.

Sous forme textuelle un SID s'exprime ainsi : S-1-5-21-3043594363-1568931741-1397882902-1009. L'entête S-1-5 exprime la version (v. 1) et l'autorité (5 = SECURITY_NT_AUTHORITY, 2 = SECURITY_LOCAL_AUTHORITY, …). Les RID < 1000 sont réservés. Les RID entre 500 et 1000 identifient des sujets connus tels que « Administrator », « Everyone », « Interactive user », etc…
Les comptes utilisateurs créés par la suite reçoivent des RID > 1000.

Principales API :
AllocateAndInitializeSid(),CreateWellKnownSid(),IsValidSid(),GetLengthSid(),LookupAccountSid(),ConvertSidToStringSid()

Descripteur de sécurité (SD)

Les objets Windows tels que fichiers, processus, etc… sont sécurisés par une structure de données attachée : le descripteur de sécurité. C'est le contenu de cette structure qui est analysée par le moniteur de référence de sécurité Windows (SRM) pour contrôler l'accès aux objets, en fonction des accréditations du sujet.

Un SD est principalement composé du :

- SID du propriétaire/créateur de l'objet
- Une liste de contrôle d'accès discrétionnaire (DACL)
- Une liste de contrôle d'audit de sécurité (SACL)

Principales API :
GetSecurityDescriptorLength (),InitializeSecurityDescriptor(), [Get]|[Set]SecurityDescriptorDacl(), [Get]|[Set]SecurityDescriptorOwner(), [Get]|[Set]KernelObjectSecurity(), [Get]|[Set]UserObjectSecurity(), [Get]|[Set]FileSecurity(),Reg[Get]|[Set]KeySecurity(),ConvertStringSecurityDescriptorToSecurityDescriptor()

Liste de contrôle d'accès (ACL)

Liste comprenant des entrées de contrôle d'accès (ACE) définissant la matrice d'accès à l'objet. Une ACE est formée de :

- SID identifiant le sujet ou le groupe de sécurité
- Masque d'accès déterminant les types d'accès possibles (lecture, écriture, exécution (dépend du type d'objet)
- Le type d'ACE (Accès autorisé, refusé)
- Différents flags définissant les règles d'héritage de l'ACE

Principales API :
GetAclInformation()

Privilèges

Si les DACL contrôlent l'accès aux objets, les privilèges contrôlent l'accès à certaines fonctions clés du système. Les privilèges sont accordés, par les politiques de sécurité, à certains groupes d'utilisateurs. Les privilèges accordés ne sont pas forcément actifs. L'application doit alors appeler AdjustTokenPrivileges(). Un privilège autorise par exemple certains utilisateurs à arrêter ou redémarrer l'ordinateur.

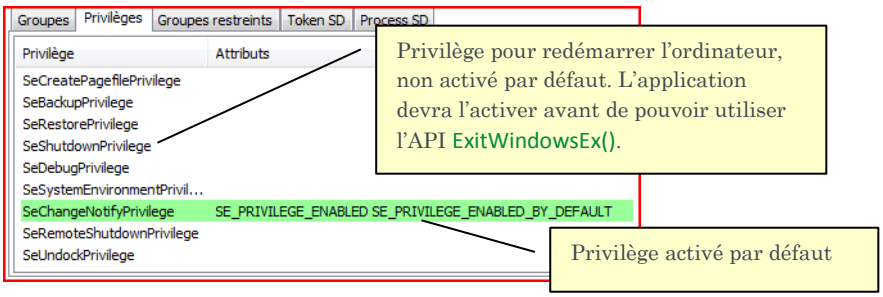


Figure 4 : Activation des privilèges vue par l'outil SecurityExplorer

Principales API :
LookupPrivilegeValue, AdjustTokenPrivileges(),

Jeton d'accès

Un jeton d'accès est créé lors du processus de login, par le sous-système d'authentification local (LSASS). Chaque processus est rattaché, à sa création, à un jeton d'accès « primaire ». Par défaut, un thread n'a pas de jeton propre. C'est le jeton de son processus qui défini son identité. Par contre, par un mécanisme d'emprunt d'identité (impersonation), un thread peut changer d'identité ou prendre l'identité d'un sujet / d'un processus client. Le thread sera alors attaché à un jeton « d'impersonalisation ». Les composants du jeton d'accès sont détaillés en annexe. La première propriété importante pour ce travail est le fait qu'un jeton primaire (le type de jeton utilisé pour créer un processus) possède un identificateur de session. Une fois le processus créé dans la session indiquée dans son jeton d'accès, il n'est plus possible d'en changer.
L'autre propriété importante est le jeton lié. En effet, nous verrons que Windows Vista créé 2 versions du jeton pour un compte privilégié, une limitée et une complète. La propriété « LinkedToken » permet d'accéder au jeton complet à partir du jeton limité et vice versa.

Principales API :
[Get]|[Set]TokenInformation(), OpenProcessToken(),

Limites de processus

Dans le monde sécuritaire idéal, les processus seraient totalement isolés entre eux. Toutefois, les applications informatiques modernes sont de plus en plus découpées et interdépendantes. Les processus doivent communiquer au sein d'un ordinateur.
Le défi pour les OS modernes est de permettre cette communication de manière sécurisée.

Différentes frontières définissent ce qui peut y être échangé :

- La ***session*** est une frontière pour les objets nommés du noyau.
- La « ***Windows Station*** » est une frontière pour le « clipboard » et les «atom tables » (échanges DDE)
- Le « ***Desktop*** » est une frontière pour les messages de fenêtre.

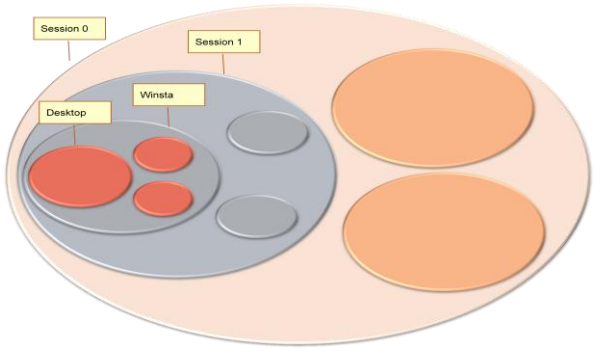


Figure 5 limites des processus

Sessions

Les premières versions 3 à 4 de la génération NT de Windows, étaient des systèmes multitâches mais mono utilisateur. La contribution de Citrix® dans l'évolution du noyau Windows vers un système multi utilisateur introduit le concept de session, qui sera intégré à Windows 2000 serveur puis à Windows XP (fast user switching).
Une session instancie un espace de noms virtuel, isolant les objets des différentes sessions. Le sous système de gestion des sessions (SMSS) créé 2 processus initiaux pour chaque nouvelle session: Winlogon et Csrss. Finalement, Explorer.exe est créé en tant que « shell » de l'interface utilisateur. Un processus est toujours créé dans la session identifiée par le champ « SessionID » de son jeton d'accès.

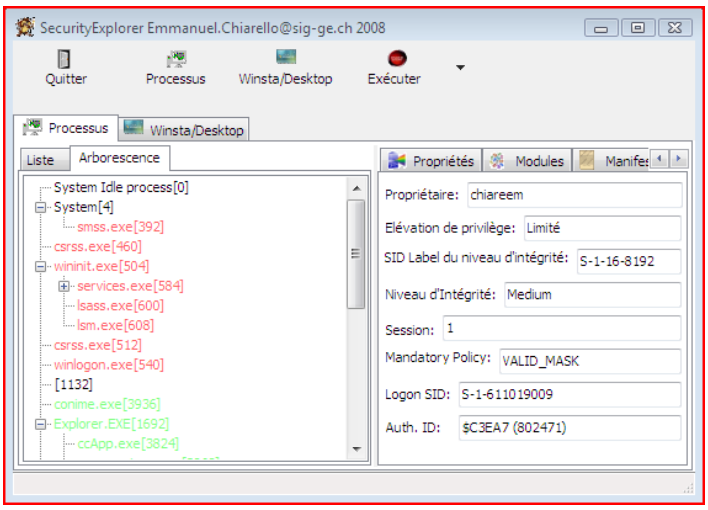


Figure 6 Arborescence des processus sous Windows VISTA

Espace de noms Windows

Le noyau Windows gère l'ensemble de ses objets (filesystems, processus, events, mutex, etc…) dans un espace de noms. Ces objets sont sécurisés (contrôle d'accès, audit, etc…).

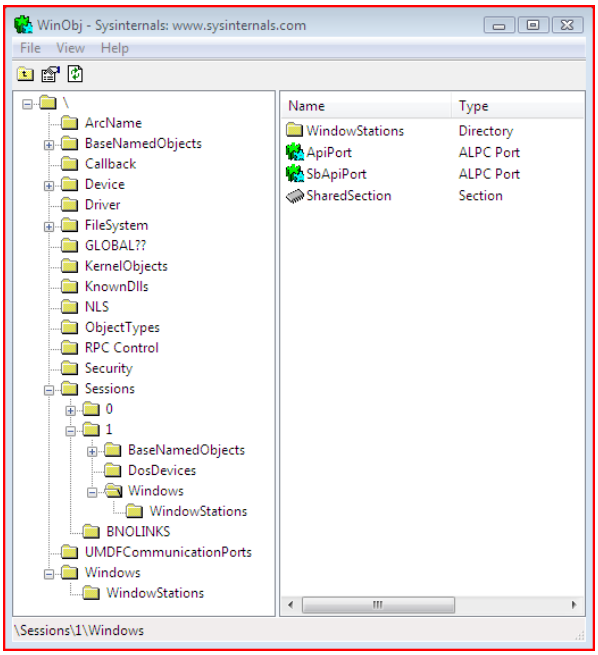


Figure 7 Visualisation de l'espace de noms Windows par Winobjs.exe (11)

Initialement sous WNT3 et 4, les « Windows stations » étaient montées directement dans le dossier \Windows.
Terminal services ajoute le dossier « \Sessions » pour héberger les objets indépendamment par session. Le dossier « \Windows » n'est plus qu'un lien symbolique vers « \Sessions*ID_de_session*\Windows ».
Avant Windows Vista, la première ouverture de session interactive (session console) est ouverte en session0, la même que celle des services. La session 0 accède directement à la racine de l'espace de noms, pas les autres. Les applications accédant des objets globaux (inter-sessions) doivent maintenant préfixer les noms des objets globaux par « \GLOBAL\ » (12) (13)

Winsta

Une fois la session créée par SMSS, une première « Window station » ou « Winsta » est créée : Winsta0. Sous Windows Vista, Winsta0 de la session 0 n'est pas interactive, contrairement aux versions précédentes de Windows. Les Winsta0 des autres sessions sont toujours interactives, c'est-à-dire attachées à une console ou à un hôte distant par RDP (Remote desktop Protocol)

Plusieurs « Winsta » peuvent être créées par session. Une « Winsta » contient un « clipboard », une « atom table » et différents « Desktop ».
Une « Atom table » est une table associant une chaîne de caractères à un mot de 16 bits. Une application place une chaîne dans une « atom table » et reçoit un identificateur lui permettant de retrouver la chaîne. Une chaîne placée dans une « atom table » est appelée « Atom name » (14). Les « atom table » sont utilisées dans plusieurs fonctions Windows comme : RegisterClipboardFormat, RegisterClass ainsi que les fonctions héritées du DDE (Dynamic Data Exchange)

Desktop

Un Desktop contient une surface d'affichage logique pour les objets d'interface utilisateur comme les fenêtres et les menus. Les messages Windows ne peuvent être échangés qu'au sein du même « Desktop ».
De même qu'une seule « Winsta » par session peut être interactive, un seul « Desktop » peut recevoir les messages d'entrée (Input desktop).

Trois « Desktop » sont créés par défaut pour Winsta0 :

- Default
- Disconnect (Screen-saver)
- Winlogon (écran de verrouillage et d'ouverture de session)

Un processus est attaché à une « Winsta » et un thread à un « Desktop ». Les « Winsta » et « Desktop » sont des objets Windows sécurisés et donc contrôlés par un descripteur de sécurité (SD). Le « Desktop » est vu comme la fenêtre parente de toutes les fenêtres applicatives.

Les fenêtres applicatives ne sont pas des objets sécurisables. L'envoi des messages de fenêtres (envoyés par Post/SendMessage()) ne peut donc pas être contrôlé.

Nous verrons plus loin que Windows Vista propose de nouveaux mécanismes pour contrôler les messages échangés entre fenêtres applicatives.

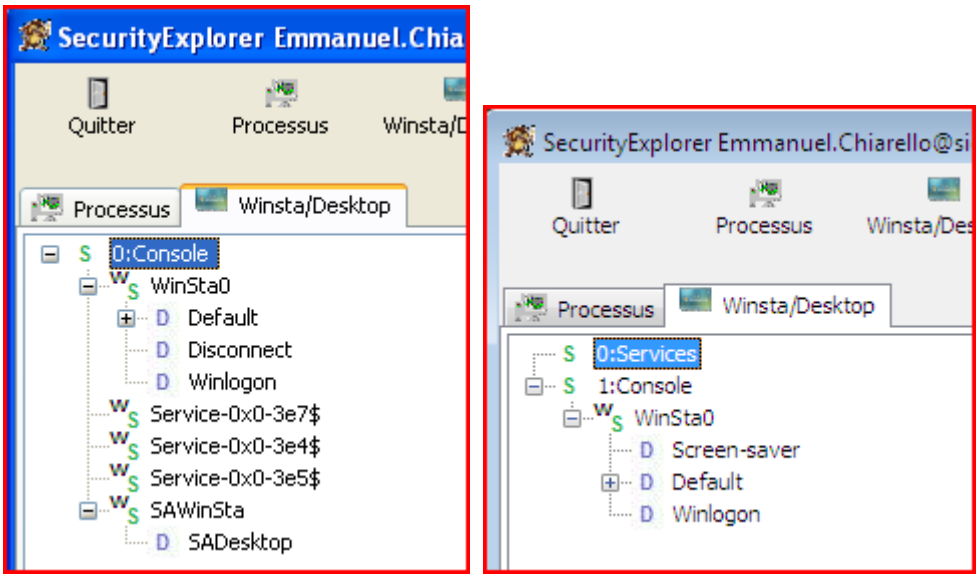


Figure 8 Arborescence des Sessions, Winsta et Desktop par le SecurityExplorer du projet, sous Windows XP (à gauche), et sous Windows Vista (à droite)

Principales API :

[Set]|[Get]ProcessWindowStation(), OpenWindowStation(), CreateWindowStation(), SetProcessWindowStation(), [Set]|[Get]ThreadDesktop(), CreateDesktop(), OpenDesktop(), OpenInputDesktop(), SwitchDesktop()

NOUVEAUTES WINDOWS VISTA

Le noyau Windows Vista est une évolution majeure par rapport aux versions précédentes, comme le montre les identificateurs de versions Windows :

- NT4 Work. 4 } Noyaux identique
- NT4 server 4
- Windows 2000 Work. 5 } Noyaux identique
- Windows 2000 server 5
- Windows XP 5.1
- Windows 2003 5.2
- Windows Vista 6 } Noyaux identique
- Windows 2008 6

On remarque que les noyaux sont à nouveau identiques entre les versions Serveur et Station de travail des versions Vista/2008.
L'objectif affiché de Windows Vista est d'augmenter sa résistance aux attaques (logiciels malveillants, virus, Trojan, etc…)

Pour réaliser cet objectif, Vista vise particulièrement :

- L'élévation de privilège
- Les « buffers overflows »
- Les « Shatter attack »

L'élévation de privilège est généralement le but recherché par un programme pirate. Les « buffers overflows » et les « shatters attacks » sont des moyens de la réaliser.
Les mécanismes de protection décrits ci-après sont des réponses spécifiques à ces attaques, mais Windows Vista en propose d'autres, génériques, tels que l'UAC et le contrôle d'intégrité obligatoire.

Les « buffers overflows », permettent l'exécution arbitraire de code par des mécanismes de débordement de buffers mal dimensionnés ou non contrôlés.
Windows Vista utilise intensément les annotations SAL des nouveaux compilateurs C++ qui, en plus du bannissement des fonctions non sûres de la librairie C, contribuent à éviter 60% des dépassements de buffer (15 p. 2). Le mécanisme DEP, introduit dans Windows XP SP2, empêche quand à lui l'exécution de zones mémoires dédiées à des données, comme le tas global ou local.

Les « shatter attack » profitent de l'absence de mécanisme de sécurité sur les objets USER (objets d'interface tels que les fenêtres applicatives). En effet, seuls les objets Kernel sont sécurisés et donc protégés par SRM (Security Reference Monitor). Ce choix s'explique probablement par un souci de performances (éviter un changement de contexte vers le noyau à chaque message reçu).
Chris Paget décrit en 2002, le principe de ces attaques (16) et les risques liés au partage d'un même Desktop par des processus de niveaux de privilèges différents.
L'isolation de la session 0 limite ce risque pour les services, mais c'est UIPI (User Interface Privilege Isolation), mécanisme de contrôle d'intégrité pour les objets USER32, basé sur les labels d'intégrité des processus qui offre une protection contre l'envoi de messages « dangereux » ainsi que l'injection de threads ou encore les hooks de messages entre processus de niveau d'intégrité différents.

Contrôle d'intégrité

L'innovation la plus intéressante au niveau de la sécurité dans Windows Vista concerne l'introduction d'un mécanisme de contrôle d'intégrité (MIC pour Mandatory Integrity Control). Microsoft cherchant probablement à prétendre, à terme, à un niveau TCSEC B1 (17 p. 3) pour Windows. Les contrôles d'accès discrétionnaires ne protègent pas suffisamment le système. Un contrôle d'intégrité obligatoire est devenu nécessaire (18 p. 21) . MIC vise à garantir l'intégrité du système, en complément au contrôle d'accès discrétionnaire. Un code inconnu, potentiellement malicieux, téléchargé d'Internet et **exécuté sous un compte utilisateur légitime**, doit être empêché de modifier l'état du système, de modifier les fichiers de données utilisateur ou de manipuler le comportement d'autres applications (19).
Les objets (ou ressources) sont désormais marqués d'un label indiquant le niveau d'intégrité de l'objet. Le label d'intégrité est intégré dans la SACL du descripteur de sécurité de l'objet à protéger. Le label d'intégrité est un SID. Il s'intègre ainsi naturellement aux descripteurs de sécurité existants. Le masque d'accès de l'ACE est utilisé pour indiquer les politiques d'intégrité. Les labels obéissent aux mêmes règles d'héritage que les ACL, définies dans le champ « flags » de l'ACE. Une modification du SRM (Security Reference Monitor) permet la prise en compte des labels d'intégrité en plus des ACL existantes, lors du contrôle d'accès.

MIC n'implémente pas le modèle « Biba ». Microsoft pense que ce modèle n'est pas adapté au niveau d'interactivité élevé et l'usage universel de Windows (15 p. 40). Le design du contrôle d'intégrité de Windows vise premièrement à contrecarrer les mécanismes d'élévation de privilèges dans les environnements Windows hautement collaboratifs (19).
En fait, MIC cherche uniquement à protéger l'intégrité du système alors que le modèle « Biba » vise l'intégrité au sens général, y compris des données.

La différence la plus importante se trouve au niveau de l'axiome « Biba » d'intégrité simple NO_READ_DOWN qui n'existe pas dans MIC (considéré comme un « Privacy control » par Steve Riley (20)). Par contre, MIC implémente la politique NO_READ_UP en plus des axiomes d'intégrité d'écriture et d'invocation du modèle « Biba ».

Attention, de nombreux documents mentionnent un contrôle d'intégrité NO_READ_DOWN pour MIC, y compris dans des documents Microsoft, par exemple (18 p. 31).

Pour une description complète du design de MIC, on consultera (21).
Et pour un avis plus nuancé, l'intéressant débat sur le blog de Steve Riley (20)

Code	Politique d'intégrité Windows	Axiome « Biba »
NR	No-read up	-
NW	No-write up	Axiome d'intégrité *
NX	No-Execute Up	Axiome d'intégrité d'invocation
-	-	Axiome d'intégrité simple

Tableau 3 Equivalence entre politiques d'intégrité Windows et axiomes du modèle Biba

Marquage des objets :

Windows Vista ne labellise pas systématiquement tous les objets créés. Par contre, tous les objets « sécurisables » sont soumis au contrôle d'intégrité, à travers le label implicite MEDIUM considéré en l'absence de label explicite.

Les objets suivants sont marqués d'après le niveau d'intégrité du jeton d'accès utilisé pour leur création (intégrité de leur créateur):

- Les processus
- Les threads ayant un jeton d'emprunt d'identité
- Les jobs (containers de processus)

D'autre sont marqués suivant les flags d'héritage des ACL, avec le label d'intégrité du container dans lequel ils sont créés :

- Les dossiers et fichiers
- Les clés de registre

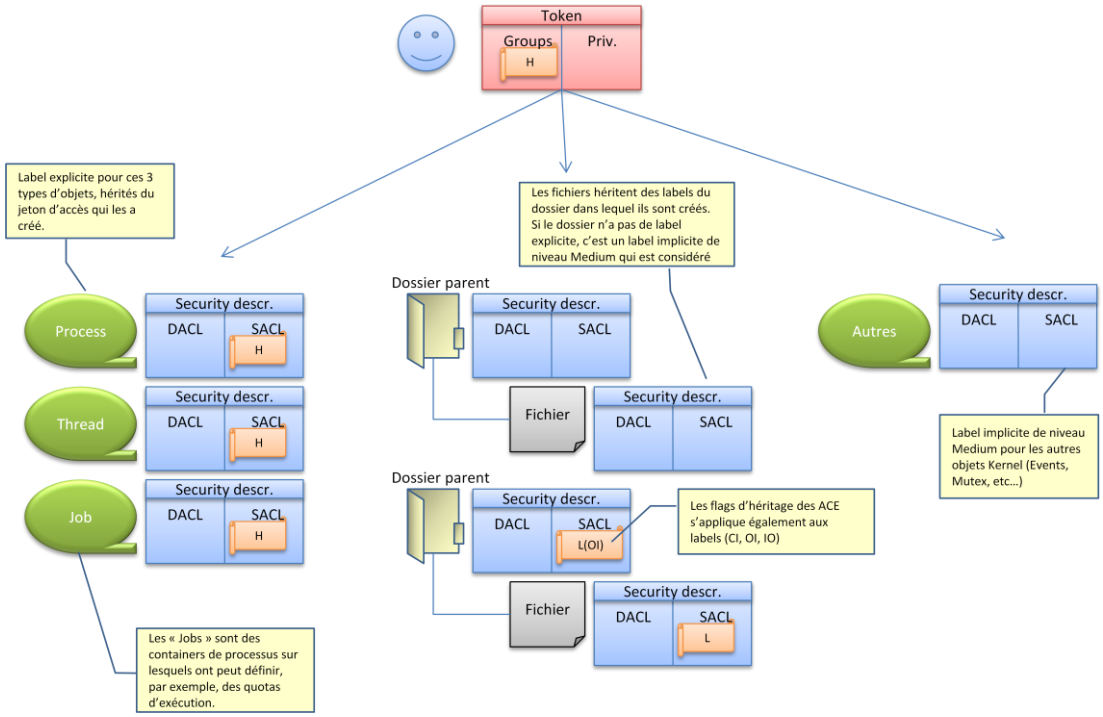


Figure 9 Labels d'intégrité à la création d'objets « Kernel » (21)

Les autres objets tels que les objets de synchronisation (Events, Mutex, …), les objets IPC (pipes, FilesMapped, …) n'ont pas de labels explicites et sont donc toujours considérés comme de niveau « Medium ». Mais le processus créateur peut attribuer un label d'intégrité - dans le descripteur de sécurité transmis comme paramètre de la méthode de création de l'objet (dans la mesure où il est de niveau inférieur ou égal à celui du processus créateur)

Le moniteur de référence de sécurité (SRM) considère les labels d'intégrité en priorité par rapport aux DACL:

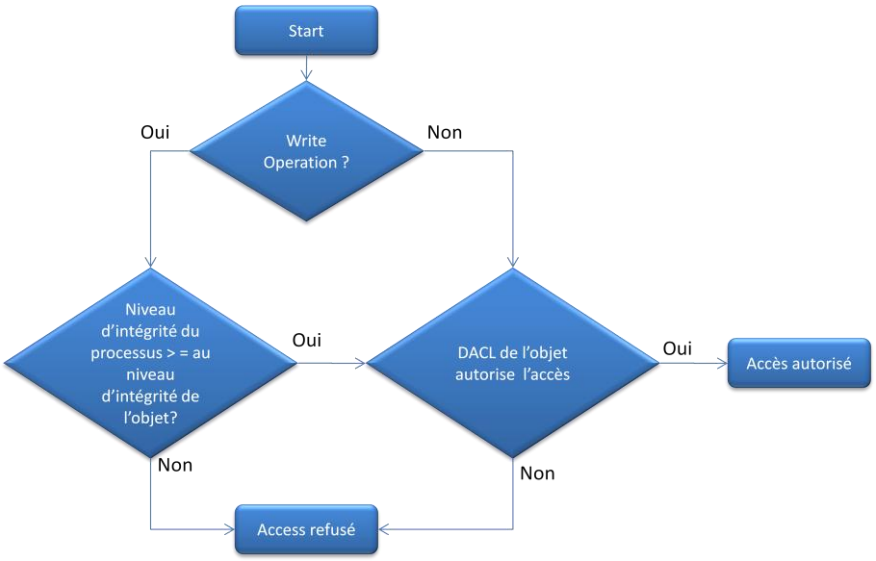


Figure 10 Diagramme simplifié déterminant l'accès à une ressource (15 p. 33)

Selon M. Howard et D. LeBlanc (15 p. 33) le contrôle d'intégrité ne s'applique actuellement qu'aux opérations d'écriture (politique NO_WRITE_UP). Toutefois certains objets comme les processus, Winstas et Desktops sont marqués de la politique d'intégrité NO_READ_UP. L'effet de cette politique sur ces objets n'est pas documenté, aujourd'hui apparemment sans effet, elle est probablement réservée pour un usage futur…

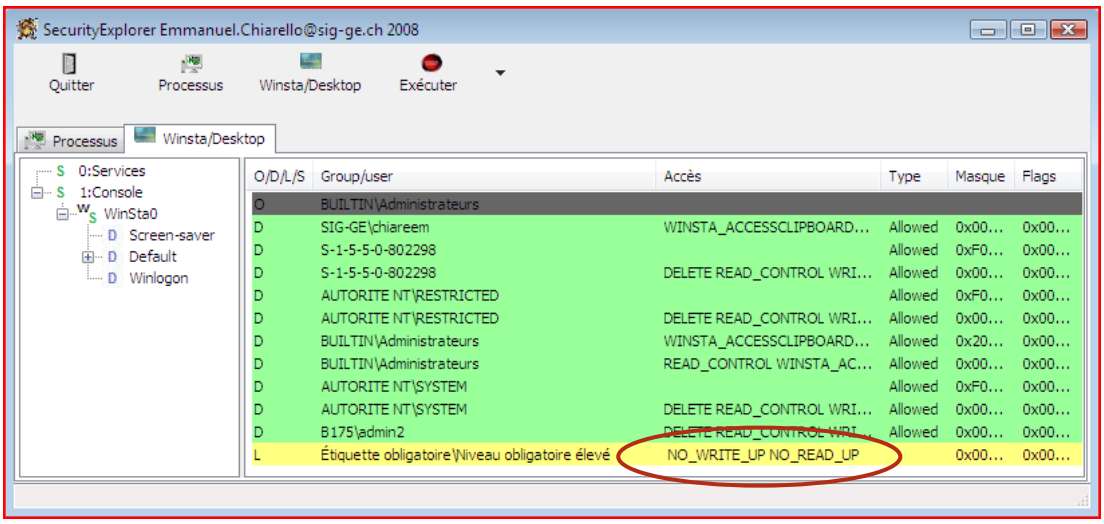


Figure 11 Politique d'intégrité NO_READ_UP appliquée aux objets Winsta et Desktop

On trouvera en annexe, la liste des dossiers et clés de registre marqués par un label d'intégrité explicite.

Politiques obligatoires :

Les niveaux d'intégrités attribués aux processus dépendent de la combinaison de 3 facteurs :

- Niveau d'intégrité marqué dans le jeton d'accès.
- Niveau d'intégrité marqué au niveau du descripteur de sécurité du fichier exécutable.
- Politique obligatoire marquée au niveau du jeton d'accès.

La combinaison de ces facteurs détermine en finalité le niveau d'intégrité du processus. Tous les jetons d'accès comportent la politique obligatoire NO_WRITE_UP. Par contre, certains d'entre eux sont marqués en plus de la politique NEW_PROCESS_MIN.
La combinaison de ces deux politiques se nomme VALID_MASK.
L'attribution des politiques obligatoires n'est pas documentée mais on peut déduire empiriquement les règles suivantes :

- Les processus avec jeton élevé (y compris les services SYSTEM) sont marqués avec la police NO_WRITE_UP … sauf : Svchost, spoolsv, SearchIndex, Dllhost, TaskEng, etc.
- Tous les autres sont marqués VALID_MASK (NO_WRITE_UP + NEW_PROCESS_MIN)

L'effet de la politique obligatoire NEW_PROCESS_MIN sur la création d'un processus dépend alors du label d'intégrité du descripteur de sécurité attaché au fichier exécutable. Ce mécanisme permet de limiter le niveau d'intégrité d'un processus exécuté à partir de code peu sûr (p.ex. fichier téléchargé d'internet marqué du niveau « Low » par Internet Explorer).

Le principe est illustré dans le schéma suivant :

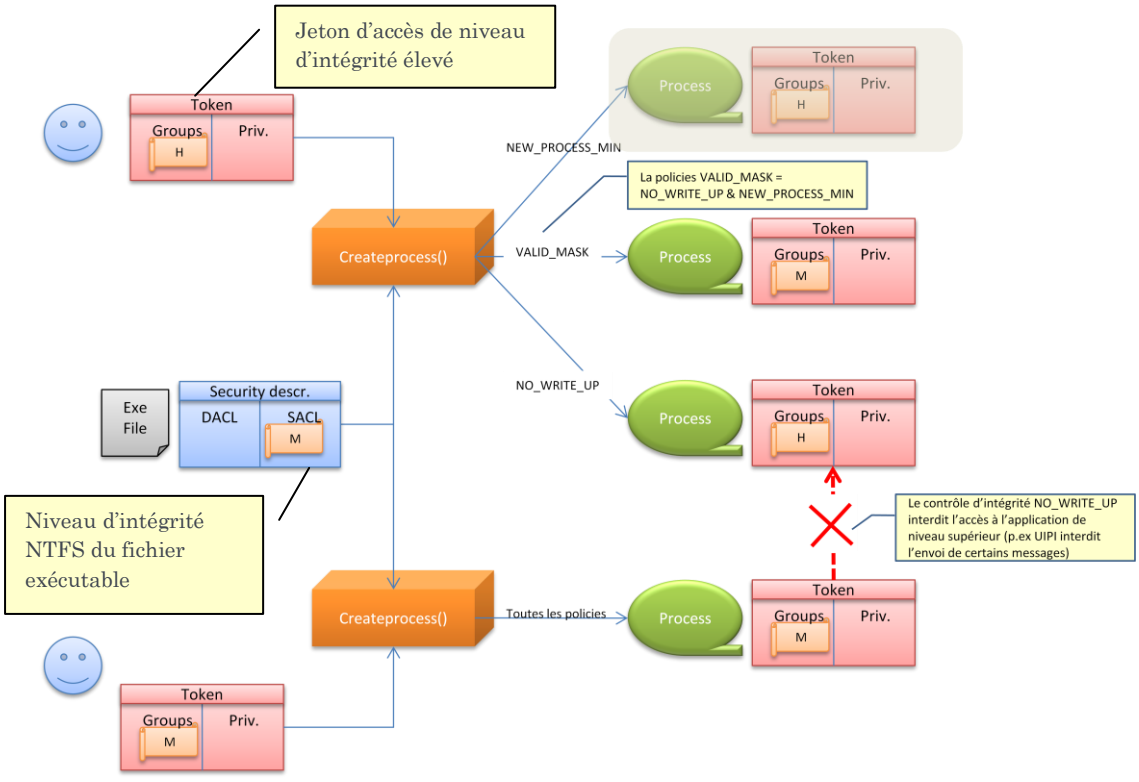


Figure 12 Effet des labels d'intégrité sur la création des processus en fonction des politiques obligatoires (21)

Contrôle des labels d'intégrité

Windows Vista assigne des labels d'intégrité à certains dossiers et clés de registre. L'identification des dossiers et clés de registre marqué par des labels d'intégrité explicites (voir *liste en annexe*) a été effectuée par l'outil **AclDump** réalisé dans ce travail. Douze dossiers et treize clés de registre ont été identifiés. Seule la racine du disque système est marquée avec un label d'intégrité HIGH. Tous les autres le sont par un label d'intégrité LOW. Les labels d'intégrité sont prévus pour être transparents vis-à-vis d'une utilisation normale et doivent rester sous le contrôle du système. A priori aucun impact pour l'installation de logiciels par télédistribution.

MIC est appelé à un rôle encore plus important dans le futur (18 p. 20)
On peut penser par exemple que d'avantage de dossiers et clés de registre seront protégés par des labels HIGH (dossier SYSTEM32, clé HKLM). De même, les niveaux implicites MEDIUM pour les objets tels que les named-pipes, Events, etc. seront peut être définis prochainement au niveau d'intégrité du jeton d'accès du processus créateur.
Par contre, il ne semble pas que Microsoft prévoie d'utiliser MIC pour protéger l'intégrité des données.

Principales API :

Pour lire ou modifier le label d'intégrité d'un objet fichier, on utilise [Get]|[Set]FileSecurity(), avec SecurityInformation = LABEL_SECURITY_INFORMATION.

Jeton limité

Un jeton limité est une copie d'un jeton original, pour lequel certains groupes ou privilèges ont été retirés. L'intérêt est de pouvoir créer un processus ou changer l'identité d'un thread, tout en limitant ses possibilités d'action. Le risque en cas de détournement du processus est ainsi limité.
Windows 2000 introduit déjà la possibilité de créer un jeton limité à l'aide de la fonction CreateRestrictedToken(). Cette fonction prend comme paramètre une liste de groupes à désactiver (voir *Figure 13*) et une liste de privilèges à supprimer.
Windows XP rend accessible cette fonctionnalité au niveau de l'interface utilisateur (explorer) en permettant l'exécution d'un processus avec un jeton restreint.

Mais Windows Vista va plus loin…

Le sous-système d'authentification LSASS a maintenant la possibilité de créer des paires de jetons liés. Un champ supplémentaire dans le jeton d'accès sert de référence vers le jeton limité/complet (GetTokenInformation() avec TokenLinkedToken).
L'API LogonUser(), qui retourne un jeton d'accès identifiant un utilisateur dont on a passé les informations de login, retournera pour un compte privilégié sous Windows Vista la version limitée du jeton d'accès.

Ainsi, 3 types de jeton existent :

- Default (jeton unique pour les comptes non-privilégiés ou systèmes. Un seul jeton créé)
- Limited (version limitée du jeton d'un utilisateur privilégié)
- Full (version complète du jeton d'un utilisateur privilégié)

Le tableau ci-dessous liste les groupes ou privilèges pour lesquels LSASS crée un jeton limité associé au jeton complet :

Groupes	Privilèges
Adminstrators	Create Token
Domain Controllers	TCB
Cert. Admins	Take Ownership
Schema Admins	Backup
Enterprise Admins	Restore
Domain Policy Admins	Debug
Power Users	Impersonate
Account Operators	Relabel
System Operators	LoadDriver
Print Operators	
Backup Operators	
RAS Servers	
Pre W2K Compat Access	
Network Configuration Operators	
Crypto Operators	

Tableau 4 Groupes et privilèges provoquant la création d'un jeton limité (22)

Les privilèges mentionnés sont retirés du jeton limité alors que les groupes sont marqués « USE_FOR_DENY_ONLY ». La raison de ce marquage au lieu du retrait pur et simple est qu'une ressource peut contenir une ACE refusant l'accès pour un groupe privilégié (p.ex. comme « domain admins »). Le retrait du groupe du jeton limité autoriserait l'utilisateur d'accéder, avec un jeton limité, à une ressource normalement interdite!

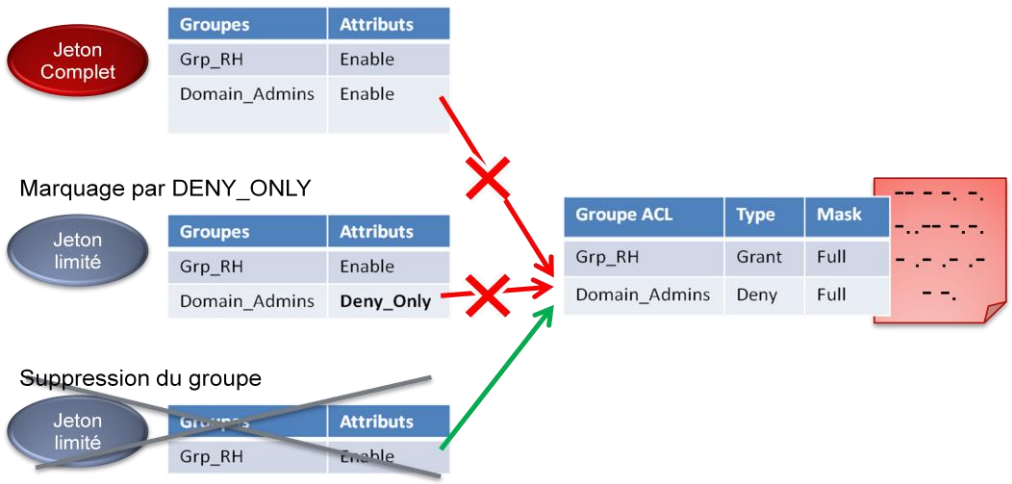


Figure 13 : Principe du marquage des groupes « privilégiés » pour refus seulement

(Dans la litérature MSDN, jeton limité = « Restricted Token » et jeton complet = « Full Token »)

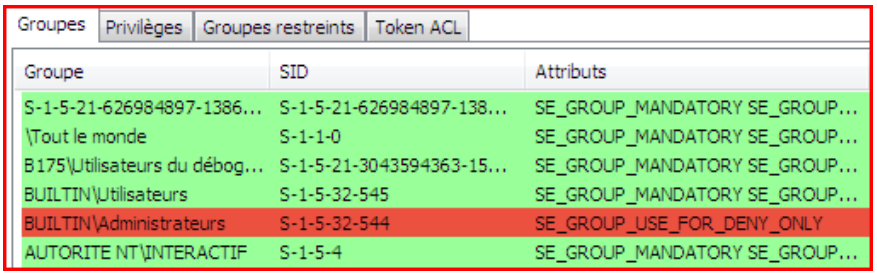


Figure 14 Jeton limité : groupe « Administrateurs » marqué pour refus seulement

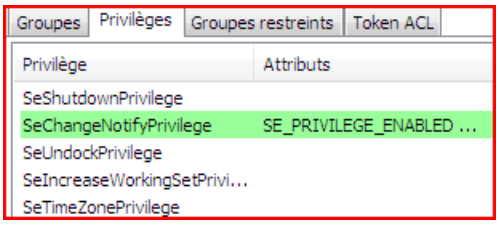


Figure 15 Jeton limité : privilèges restants dans le jeton limité

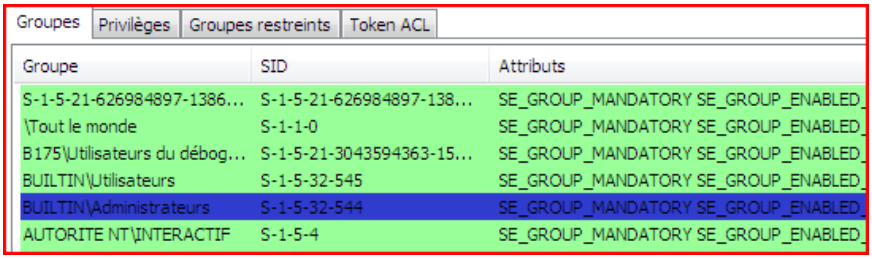


Figure 16 Jeton complet : Groupe « Administrateurs » actif

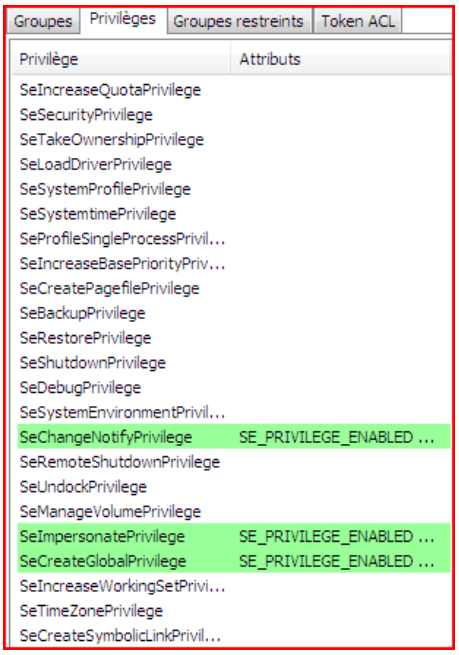


Figure 17 Jeton complet : ensemble des privilèges disponibles

Isolation des services en session 0

Les versions XP/2003 de Windows isolaient les services dans des Winsta différentes de la Winsta interactive de la session utilisateur, au sein de la session 0.
La 1ère session utilisateur partageait toutefois la même session 0 que les services.
Un service marqué pour interagir avec le Desktop est démarré avec le compte SYSTEM, dans la Winsta « Winsta0 » et le Desktop « Default », les mêmes que ceux des programmes utilisateurs interactifs de la session 0.
De même, un processus/thread peut changer de Winsta et de Desktop après sa création.
L'existence de services interactifs est un risque de sécurité important puisque les objets d'interface utilisateur ne sont pas sécurisés. Globalement, la cohabitation de processus de privilèges différents au sein d'une même session est un problème de sécurité majeur et compromet l'intégrité du système.
Windows Vista isole les processus de services dans une session 0 distincte des sessions utilisateur (la 1ère session utilisateur est alors la session 1). La session d'un processus, marquée au niveau du jeton d'accès, ne peut plus être changée pour la durée de vie du processus. Ainsi, les Winsta et Desktop interactifs sont hors de portée des services et vice-versa.

User Interface Privilege Isolation(UIPI)

Comme expliqué précédemment, le mécanisme de fenêtrage de Windows ne s'appuie pas sur des objets du noyau, mais sur des objets de la librairie USER. Il en découle l'absence de mécanismes de sécurité tels que les ACL pour limiter l'accès à ces objets. Depuis 2002 de nombreuses méthodes documentées (p.ex. (23) ont démontré la faiblesse du système de fenêtrage Windows, permettant l'élévation de privilèges par l'envoi de messages spécifiques vers des fenêtres d'applications de niveau de privilège supérieur (Administrateur ou SYSTEM).
Windows Vista corrige en partie ce problème, en isolant les services dans une session 0 indépendante. Toutefois, certains processus de privilège élevé doivent continuer à cohabiter dans la session utilisateur (p.ex. CSRSS.EXE et Winlogon.exe). C'est pourquoi Windows Vista met en place un nouveau mécanisme pour contrôler le flux de messages entre applications de niveaux d'intégrité différents.
UIPI se défini donc comme un mécanisme de contrôle d'intégrité pour le niveau USER. Il contrôle les mécanismes inter processus susceptibles d'échapper au SRM, comme l'échange de messages de fenêtres, l'injection de thread ou les crochets applicatifs (hook) qui s'insèrent dans la boucle de traitement des messages. UIPI se base sur les niveaux d'intégrité MIC des processus, pour bloquer les messages jugés dangereux (messages modifiant le comportement de l'application ou permettant l'échange de données) vers un processus de niveau d'intégrité supérieur.

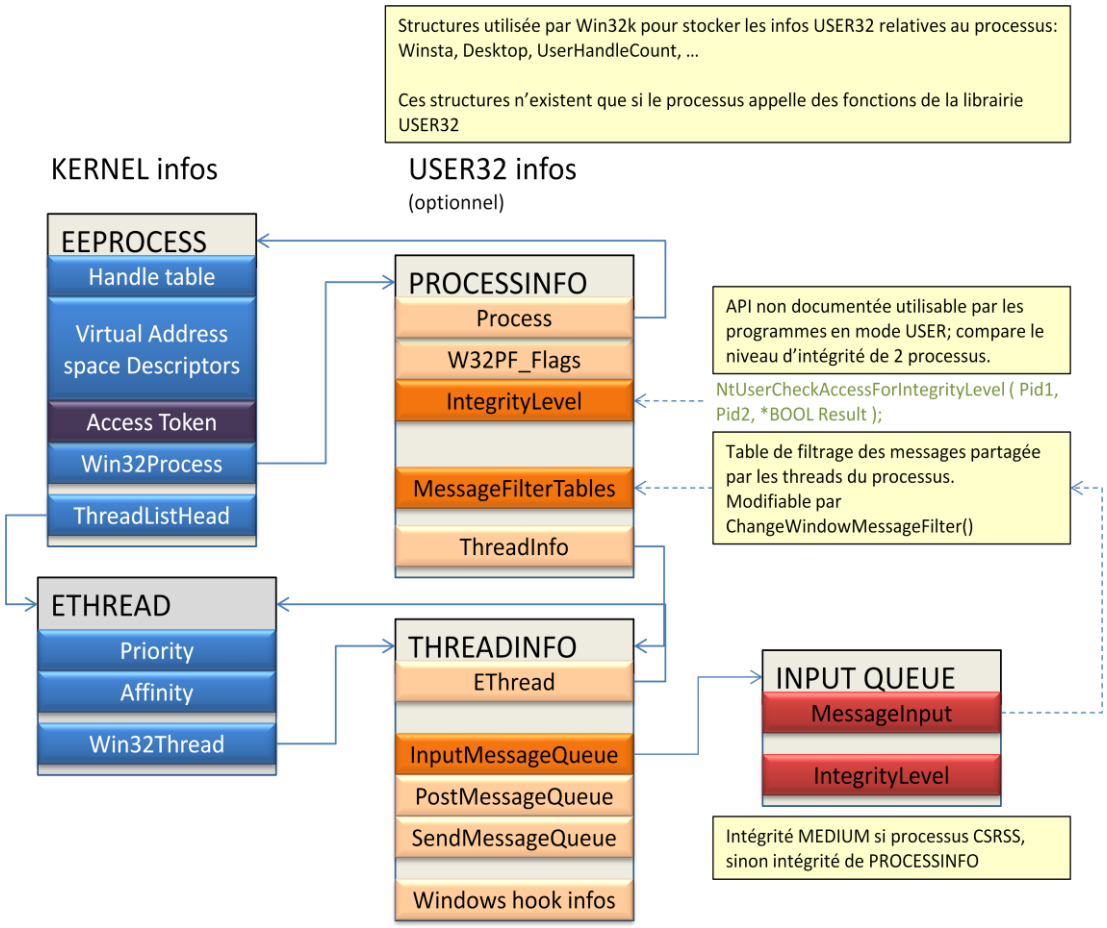


Figure 18 : Structures de données Kernel et User32 utilisées par UIPI (24)

L'outil de test (TestIntegrite.exe) réalisé lors de ce travail met en évidence le mécanisme UIPI pour le blocage de messages et l'injection de thread par CreateRemoteThread().
A noter que le filtrage par défaut des messages peut être modifié grâce à une nouvelle API : ChangeWindowMessageFilter(msg: UINT, dwFlag: DWORD). Le filtrage se définit au niveau processus et non au niveau thread (voir *Figure 18 : Structures de données Kernel et User32 utilisées par UIPI*)

Exceptions

- CSRSS (Client/server Runtime Subsystem):
  Gère les consoles sous Vista (l'essentiel des opérations du sous-système Win32 ont été déplacées dans des pilotes en mode noyau (25)). Les queues de messages consoles des différents processus sont gérées par des threads de CSRSS. UIPI défini systématiquement le niveau d'intégrité des queues de message de CSRSS sur MEDUIM. En février 2007, J. Rutkowska décrit le moyen d'envoyer des messages WM_KEYDOWN vers une console de « shell » Administrateur (26) permettant l'exécution de commandes arbitraires à partir d'un processus de niveau d'intégrité faible, partageant le même Desktop qu'une console administrative, démontrant les faiblesses d'UIPI et les probables exploitations à venir des exceptions d'UIPI, p.ex. de CSRSS (24).

- Programmes d'accessibilité :
  Les interfaces prévues pour faciliter l'accès aux personnes handicapées doivent pouvoir contourner les mécanismes UIPI (envoi de messages simulant les mouvements de la souris ou du clavier vers des fenêtres applicatives de niveau d'intégrité potentiellement supérieur). Ce genre d'application doit respecter des conditions de marquage spécifiques, résumées dans le diagramme ci-dessous :

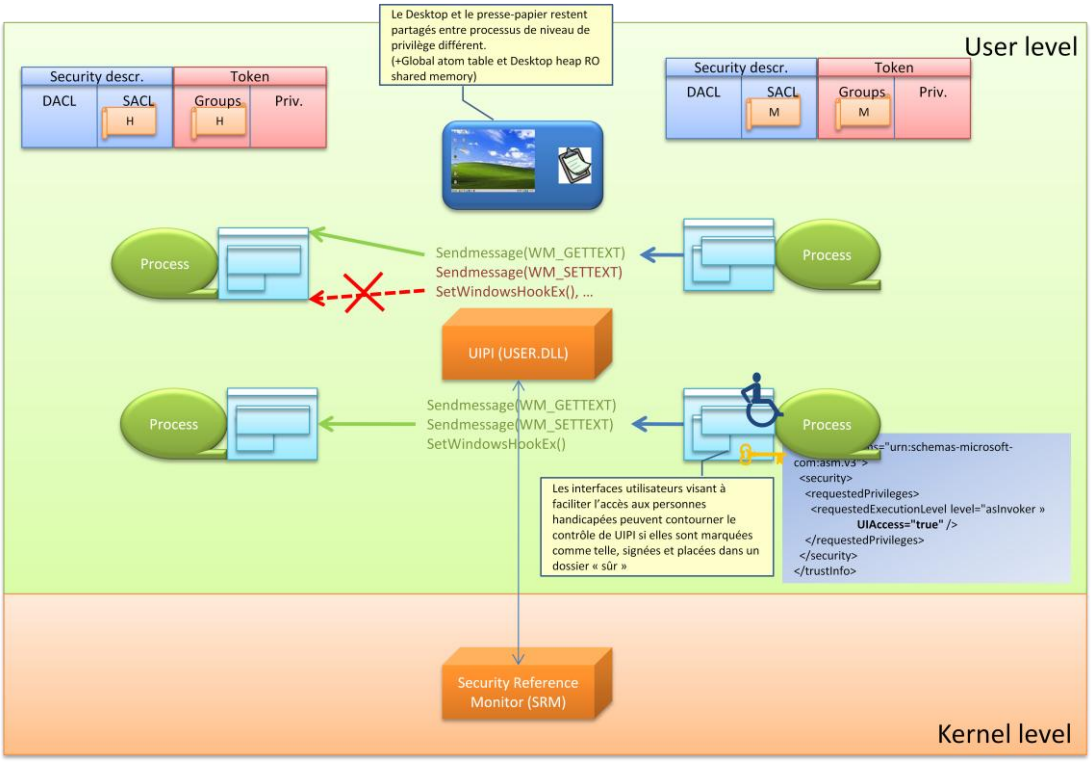


Figure 19 : UIPI pour les programmes d'accessibilité

Accessoirement, UIPI peut être désactivé, en définissant à **'0'** la valeur **'Enable UIPI'** dans la clé de registre :
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System

Attention, désactiver la police de sécurité 'EnableLUA' sous HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\ désactive également le filtrage des messages par UIPI, mais l'injection de threads reste bloquée si elle est tentée à partir d'un processus d'intégrité plus faible.

Pour une description des mécanismes internes d'UIPI, voir l'excellent travail de E. Barbosa (24)

Consentement d'élévation de privilèges

Windows Vista introduit un mécanisme de consentement utilisateur pour les demandes d'élévation de privilèges. Une application peut nécessiter une élévation de privilège au démarrage ou en cours d'exécution. La manière de contrôler l'élévation au démarrage sera expliqué au chapitre *Intégration*.

Séquence d'exécution d'une application pour un utilisateur standard (27 p. 25)

- ShellExecute() appelle CreateProcess()
- CreateProcess() vérifie si l'application nécessite ou non une élévation de privilège. Si oui, l'exécutable est alors inspecté pour déterminer son « requestedExecutionLevel », qui est fourni par le manifeste de l'application s'il existe.
- CreateProcess() retourne une erreur Win32 : ERROR_ELEVATION_REQUIRED
- ShellExecute() vérifie ce retour d'erreur et passe l'appel vers le service d'information des applications (AIS ou AppInfo) qui est en charge de demander l'élévation de privilège.

Description d'élévation de privilège (27 p. 25)

- AIS reçoit l'appel de ShellExecute() et réévalue le niveau d'exécution requis ainsi que les stratégies de groupes pour déterminer si l'élévation est permise.
- Si le niveau d'exécution requiert une élévation, AIS affiche la fenêtre de consentement.
- Dès que l'utilisateur donne son consentement, AIS récupère le jeton complet associé à l'utilisateur.
- AIS appelle CreateProcessAsUser() avec le jeton complet et crée un processus avec un niveau de privilège élevé.
- Le nouveau processus élevé est rattaché à l'arborescence du processus appelant.

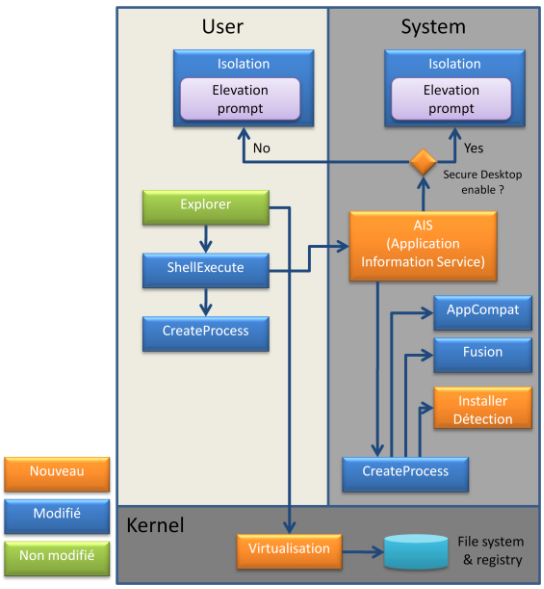


Figure 20 : Mécanise de lancement d'application sous Windows Vista (27 p. 25) et (28 p. 30)

Le seul moyen d'élever un processus consiste à créer un nouveau processus avec un jeton complet. Un processus existant ne peut être élevé car on ne peut pas changer le jeton d'un processus après sa création.

Pour respecter les relations d'héritage des handles, Windows Vista permet de définir le processus parent et les « handles » hérités pour CreateProcessAsUser() .

Virtualisation

Finalement et en marge des mécanismes de sécurité mais faisant partie du « paquet » UAC, Windows Vista introduit un mécanisme limité de virtualisation des applications. Ce mécanisme corrige certains problèmes de compatibilité applicative introduits avec la limitation des comptes administrateurs d'UAC.
Cette virtualisation consiste à intercepter les lectures/écritures ayant comme destination des dossiers ou clés de registre machine, nécessitant des droits privilégiés et à les rediriger vers des zones utilisateur spéciales. Ce mécanisme permet à certaines applications ne respectant pas les principes de développement Windows (programme « logo Windows XP » ou Vista (29)) de fonctionner sous Windows Vista sans droits administrateurs. La virtualisation de Windows Vista nous intéresse dans le cadre des télédistributions d'applications puisque certains fichiers/clés de registres applicatifs risquent de ne pas être accédés depuis leur emplacement d'origine. A remarquer que la virtualisation d'une application est conditionnée par le marquage du niveau d'exécution requis du manifest de l'application (voir chapitre *Fichier Manifest*)

La virtualisation Windows Vista est décrite en détail dans (28 p. 18)

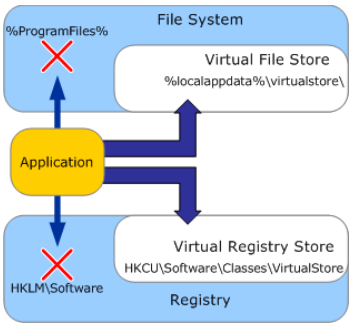


Figure 21 : Principe général de la virtualisation (28 p. 18)

Redirection des dossiers :

Les dossiers redirigés pour les applications virtualisées sont %ProgramFiles%, %ProgramData% et %SystemRoot% à l'exclusion de certains sous-dossiers. Les fichiers de type exécutable (.exe, .bat, .scr, .vbs) sont exclus de la virtualisation. D'autres exclusions peuvent être définies dans la clé de registre : HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Luafv\Parameters\ ExcludedExtensionsAdd

Le mécanisme de virtualisation des fichiers est implémenté dans le pilote Luafv.sys sous forme de filtre du système de fichiers Ntfs.sys :

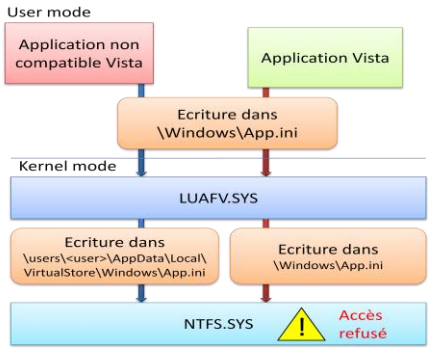


Figure 22 : Virtualisation des écritures de fichiers (30).

Redirection des clés de registre

L'implémentation de la redirection des clés du registre est légèrement différente de celle des fichiers. La plupart des clés de HKEY_LOCAL_MACHINE\Software sont redirigées, sauf certaines exceptions comme .\Microsoft\Windows, \Microsoft\Windows NT, \Classes.

Seules les clés susceptibles d'être modifiées par les applications et ne posant pas de problèmes de compatibilité sont redirigées. La redirection s'effectue vers la clé utilisateur HKEY_ CURRENT_USER\Software\Classes\VirtualStore. Jusque là, le mécanisme est semblable à celui de la virtualisation des fichiers. Par contre, au lieu de maintenir une liste des exceptions comme le fait LUAFV.SYS, le statut de virtualisation des clés de registre est enregistré comme flag dans la clé elle-même. Ce marquage ne s'applique qu'à la ruche HKLM\Software.

Flags	Signification
REG_ KEY_DONT_VIRTUALIZE	La clé n'est pas virtualisée
REG_KEY_ DONT_SILENT_FAIL	L'effet de ce flag n'est pas clairement documenté
REG_KEY_ RECURSE_FLAG	Les nouvelles sous-clés héritent les flags de virtualisation de la clé parente.

Tableau 5 : Flags de virtualisation du registre HKLM\Software

MSDN ne donne aucune information sur la valeur de ces flags ni sur les API pour les accéder ou les modifier.

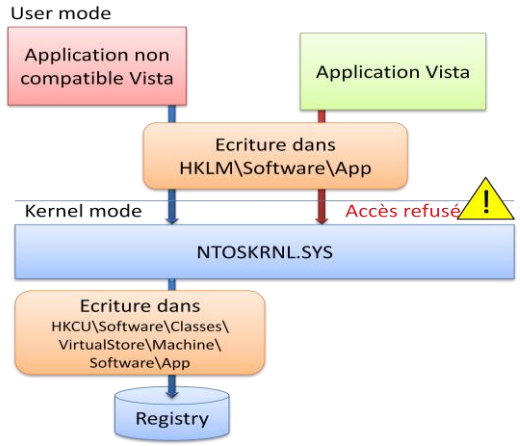


Figure 23 : Virtualisation des écritures dans le registre (30).

L'adaptation des permissions standards des dossiers et clés de registre est une activité relativement courante du déploiement d'applications en entreprise, où les droits administrateurs ne sont en général pas octroyés aux utilisateurs.
Dans ce sens, la virtualisation des applications de Windows Vista peut faciliter le déploiement des applications qui ne tiennent pas compte des restrictions liées aux comptes utilisateurs (non administrateurs).

Développement

Conformément au cahier des charges, le développement s'est fait à partir de l'IDE Turbo Delphi 2006 de Borland® (formellement Codegear®) donc avec le langage Pascal objet.

Les intérêts de Turbo Delphi pour ce projet sont :

- Compilation Win32 (non .NET)
- Objet (hiérarchie de classes, d'interfaces)
- Développement rapide des interfaces graphiques (VCL : Visual Component Library)

Et les inconvénients:

- Nécessité de traduire les entêtes des fonctions et structures des API Windows du langage C vers Delphi. Heureusement, une traduction équivalente a toujours été possible.
- Pas de compilateur 64 bits.

OUTILS

La première étape de développement a consisté à découvrir les mécanismes de sécurité Windows Vista par l'étude des API de sécurité et la construction d'outils d'exploration de ces mécanismes.

Explorateur des attributs de sécurité de processus

L'outil SecurityExplorer, inspiré de l'excellent « Process explorer » de Sysinternals (31), a permis d'expérimenter les API Windows relatives à la gestion des processus, jetons d'accès et ACL. Il permet la visualisation détaillée des attributs de sécurité de différents objets Windows comme les processus, les threads, les Winsta et Desktop, les jetons d'accès. Au niveau des processus, il affiche le fichier manifest embarqué et les jetons d'impersonalisation liés aux threads. Loin d'être un simple clone de « Process Explorer », il lui est sur bien des aspects complémentaire.

Les 2 principales classes mise au point ici sont les classes TProcess et TUserToken :

- La classe TProcess comprend par exemple des méthodes de création de processus sous un compte utilisateur donné (constructeur `CreateAsUser()`) ou avec un jeton d'intégrité faible (`CreateRestricted()`)
- La classe TUserToken comprend quand à elle, les méthodes et propriétés de manipulation des jetons. La propriété `RestrictedToken()` permet par exemple de créer un double restreint d'un jeton (Comme le fait UAC pour un compte privilégié). Les privilèges sont retirés et les groupes de sécurité marqués pour refus seulement. Ce jeton peut être utilisé pour « impersonnalier » un thread d'écoute serveur et réduire la surface d'exposition au risque.

Cet outil s'est révélé essentiel, pour expérimenter la création de processus avec jeton élevé. De même il a permis la validation des mécanismes de « hardening » du service d'installation (voir chapitre *Risques de sécurité*).

AclDump

Cet outil a été réalisé dans le but d'identifier les dossiers et clés de registre marqués par des labels d'intégrité explicites. Il a permis l'adaptation des classes existantes de gestion des descripteurs de sécurité (unité ACL2.pas) pour l'accès aux labels d'intégrité.

TestIntegrity

L'étude des mécanismes d'UIPI (voir chapitre *User Interface Privilege Isolation(UIPI)*) s'est faite en partie avec cet outil, qui permet de vérifier les mécanismes de blocage entre applications de niveaux d'intégrités différents.

TestDEP

Outil de test pour les API DEP (voir chapitre *DEP (Data Execution* Prevention)) nouvelles dans Vista SP1 et vérification de l'efficacité de ce mécanisme à travers la simulation de dépassement de buffer. Le code binaire du buffer overflow a été trouvé sur un site générateur de code pour exploit (32) et mentionné dans (15) (exécution de « calc.exe » à partir d'un buffer de données auquel on transfert l'exécution).

RunAs

Cet utilitaire permet l'exécution d'un programme interactif, à l'aide du service d'installation décrit au chapitre *Service d'installation*, sous un compte administrateur, dans la session de l'utilisateur et sans demande de consentement UAC.
Il a également été utilisé dans la phase de mise au point du service, pour exécuter sous le compte SYSTEM le service d'installation en mode console (sans passer par le gestionnaire de service).
Ce programme peut servir de base pour un programme automatisant l'élévation et remplaçant le RunAs.exe de Windows. Toutefois, j'ai considéré qu'il s'agissait là d'une utilisation détournée du service d'installation, contournant les mécanismes de sécurité Vista et n'ai donc pas souhaité fournir un tel outil dans le cadre de ce travail.

SERVICE D'INSTALLATION

Architecture

L'une des conclusions importante de l'étude des mécanismes de sécurité de Windows est qu'un processus Win32 ne peut pas changer de session. En effet, on peut définir la session cible dans un jeton d'accès et utiliser celui-ci pour créer un processus dans une session donnée, mais une fois le processus créé, on ne peut plus lui changer son jeton d'accès ni donc sa session.
Ceci implique que le service d'installation (démarré en session 0) ne peut agir directement dans les autres sessions.
Le mécanisme d'installation nécessite donc à priori **3 processus distincts** :

- Le processus client, exécuté sans privilèges particuliers, ordonnance les séquences d'installation.
- Le service d'élévation, exécuté avec le compte SYSTEM en session 0, crée un processus d'installation serveur dans la session du processus client, sous un compte administrateur.
- Le processus d'installation serveur, créé par le service d'installation dans la session utilisateur, sous un compte administrateur (et ayant accès aux ressources des serveurs du domaine).

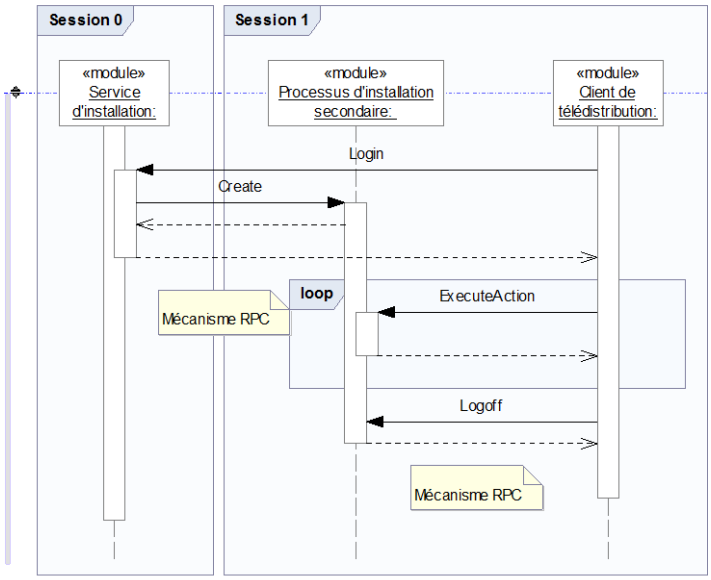


Figure 24 Principe d'exécution du service d'installation

En fait, le service d'installation et le processus d'installation ont beaucoup de code en commun. Ils ont donc été réalisés dans le même fichier exécutable, exécuté sous deux modes différents, suivant les paramètres de la ligne de commande. Au niveau du modèle de communication entre le processus client et les processus serveurs, Microsoft propose trois modèles. Le modèle qui nous intéresse se nomme **« back-end service model »** (28 p. 58), le seul qui permette l'exécution de processus élevés sans consentement UAC.

Les autres modèles possibles pour une application de type « Administrator-only » sont :

- Admin broker model
- Admin COM model

Au niveau des classes de communication client-serveur, j'ai choisi une combinaison des modèles Client-Serveur et Forwarder-Receiver) (33), baptisé « client-serveur symétrique ». Le but de ce modèle est de permettre l'appel synchrone de procédures par le client (modèle client-serveur) et la notification asynchrone par le serveur, de messages de progression et d'installation (modèle Forwarder-Receiver).
Dans le service d'installation initial, les notifications étaient envoyées au client comme messages de fenêtres par SendMessage(). Bien que UIPI ne les bloque pas (car provenant d'un processus de niveau d'intégrité supérieur), il a semblé plus opportun d'unifier les mécanismes de communication entre le client et le service d'installation, permettant par ailleurs la propagation des notifications aux appels distants (rappel : les messages Windows sont limités au desktop courant)

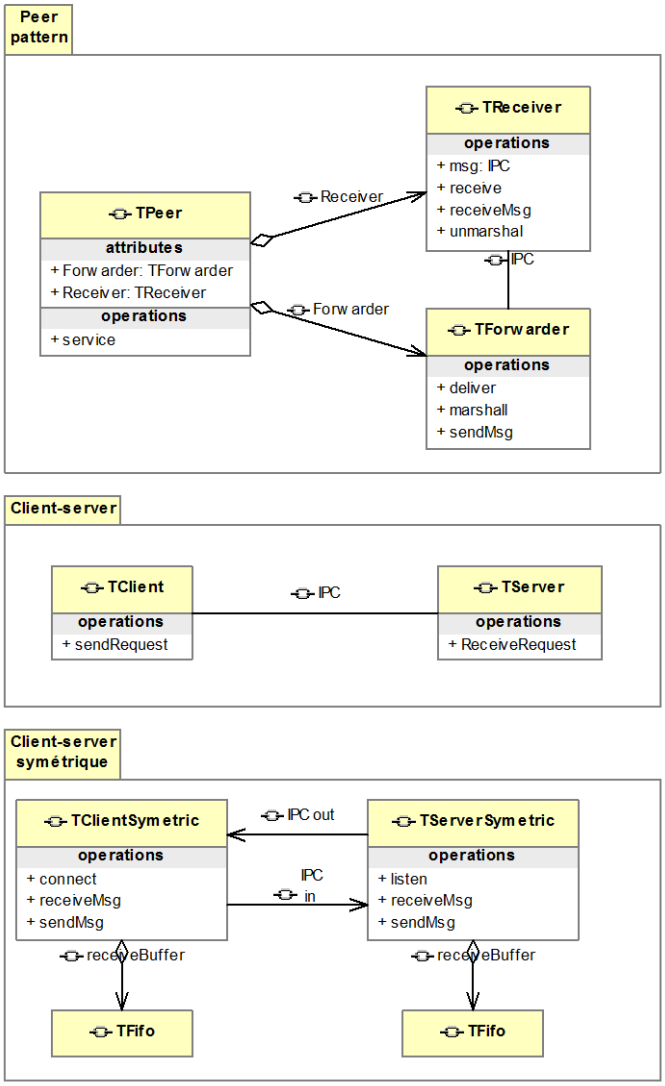


Figure 25 : Pattern « Client-serveur symétrique » utilisé dans le mécanisme RPC

L'ajout d'un tampon de lecture (TFifo) au modèle, permet de libérer rapidement le thread de lecture du Pipe. La classe TFifo implémente un tampon circulaire, protégé contre les accès concurrents (objets de synchronisation events et mutex) et sert d'interface entre le thread de lecture du pipe et le thread de traitement.

Communication IPC

Le service d'installation SIGTL initial était implémenté sous forme de composant COM (Component Object Model). C'est le programme client qui, par l'intermédiaire du gestionnaire des services (SCM), démarre le service lorsqu'il en a besoin.

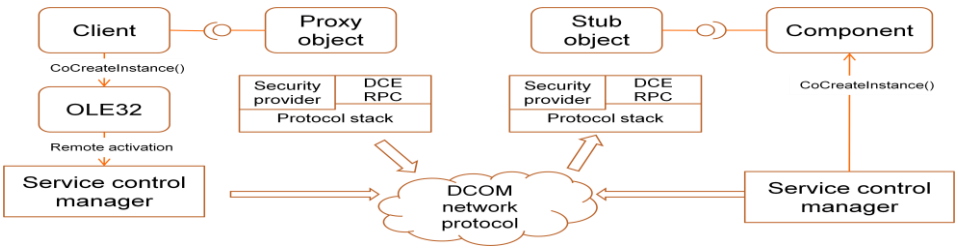


Figure 26 Mécanisme d'instanciation distant DCOM (34)

Avec COM, le service est démarré au besoin. C'est est un avantage en termes d'économie des ressources système, mais induit un délai de connexion de quelques secondes pour le démarrage et l'arrêt du service. De plus, le fait que le serveur COM soit démarré comme service système empêche toute exécution multiple. Il ne peut y avoir qu'un seul client connecté à un même serveur « service COM ». Cette limitation pose problème dans certaines situations, où le service d'installation pourrait servir à différents processus en parallèle. Par exemple lors de l'ouverture de session, le script de login doit sérialiser différentes tâches telles que l'installation de logiciels, la connexion d'imprimantes réseau et la synchronisation de fichiers systèmes (à partir d'une référence unique) ce qui allonge le temps d'ouverture de session. Ces limitations en plus du problème de blocage de communication COM sous Vista, mentionné dans l'étude du cahier des charges m'ont poussé à étudier d'autres mécanismes pour les appels RPC utilisables avec le modèle « back-end service »

Le choix du mécanisme de communication client-serveur est également déterminé par :

- Les possibilités offertes par l'IDE Delphi
- Le modèle d'architecture

Les principaux mécanismes IPC possibles sont listés dans le tableau ci-dessous :

Système	Description	Analyse
RPC	Remote procedure Call. Framework pour l'appel de procédures entre processus, intégré à Windows.	Idéal mais impossible sous Delphi. Nécessite un compilateur IDL / Delphi et/ou une complexité non justifiée pour ce projet.
Named-pipe	Tubes de communication nommés, semblable à des fichiers.	Objets sécurisables de bas niveau, accessibles à distance par le réseau.
Winsock	Communication par TCP ou UDP	Communication de bas niveau non sécurisée par ACL (possible sur Vista SP1 avec SSTP ?) (35)
COM	Extension objet des RPC Microsoft	Les restrictions de sécurité vis-à-vis de COM se durcissent à chaque version de Windows, rendant la maintenance du système couteuse. Certains aspects de COM/DCOM sont obscurs, notamment pour des serveurs COM sous forme de services.
Mémoire partagée	Zone mémoire partagée entre processus (FileMapping)	Objet de bas niveau sécurisé, performant mais pas utilisables à distance. La concurrence d'accès doit être gérée.

L'implémentation du mécanisme RPC sera basée sur les Named-pipe en raison du support natif de l'authentification, de la sécurité par ACL et de l'utilisation distante possible.

Mécanisme RPC par named-pipes

Les named-pipes sont des mécanismes de communication inter-processus (IPC) biens connus. Il s'utilisent comme des socket ou des fichiers ; on écrit des données à une extrémité du pipe avec write() et on les lit à l'autre extrémité avec read(). Windows Vista ajoute de nouvelles API en rapport avec les Named-pipe, comme la possibilité d'identifier le processus ou de la station cliente.

Toutefois, cette apparente simplicité masque quelques difficultés :

- Synchronisation des IO (risque de blocage inter thread/processus)
- Sécurisation des pipes (identification du client et contrôle d'accès)

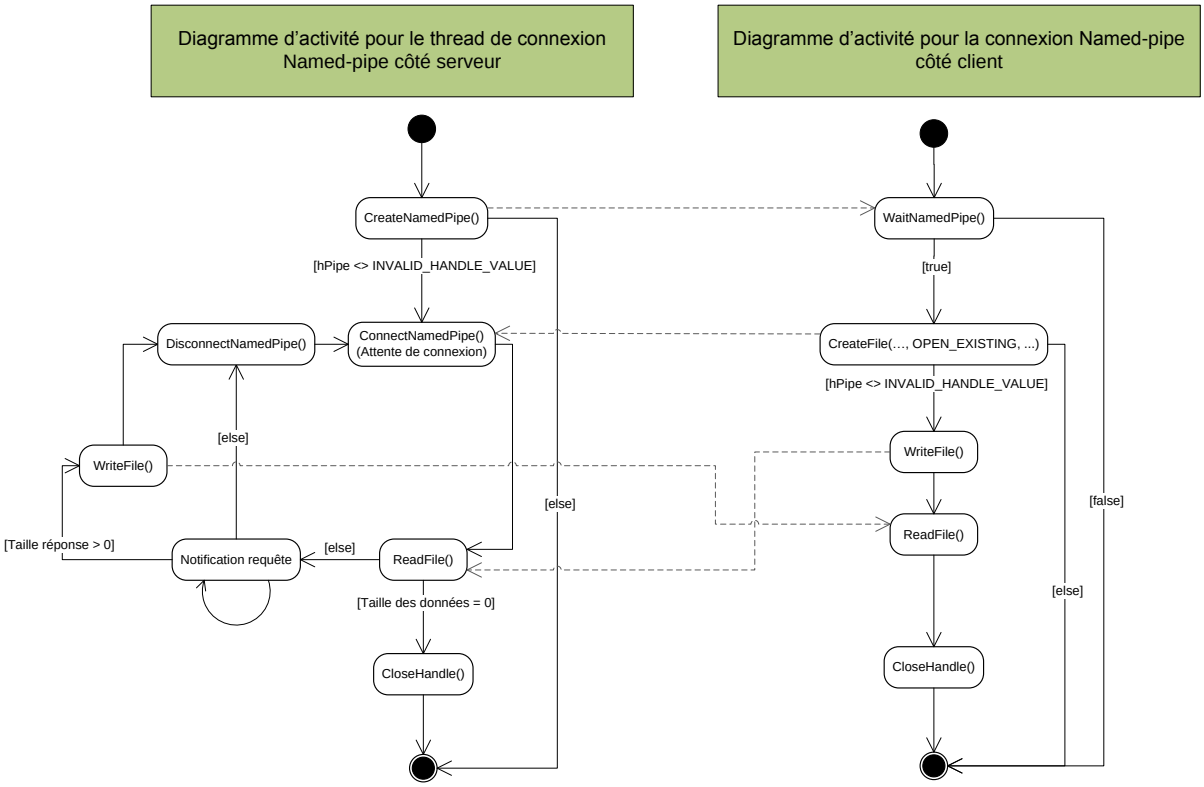


Figure 27 : diagrammes d'activité des mécanismes de base clients et serveurs par named-pipe

Framework RPC

Le framework RPC basé sur les named-pipes a été construit de manière a être totalement agnostique vis-à-vis de son utilisation concrète. Les Named-pipe permettent l'échange de flux de données entre 2 processus client et serveur. Les utiliser pour du RPC, nécessite de sérialiser les appels de méthodes distantes. Sérialiser évoque assez naturellement XML. Dans un premier temps, les tests ont été réalisés avec des transferts synchrones de messages structurés (record Delphi équivalents aux structures du C). Par la suite, le mécanisme RPC a évolué vers un échange asynchrone de flux XML « full duplex ».

Les principaux avantages de ce framework RPC sont :

- Souplesse et évolutivité grâce à XML
- Accès sûr aux données (évite le risque de débordements de buffer de taille fixe)
- Validation des messages reçus par un schéma XSD
- Notification asynchrone du client au cours d'un appel RPC
- Utilisation des IO asynchrones, plus complexes mais permettant un meilleur contrôle des éventuels blocages inter-threads.

L'implémentation concrète est simplifiée au maximum. Ainsi, la propagation des exceptions serveur vers le client est transparente et ne nécessite pas de code particulier. Dans un cas simple, l'implémentation du Stub serveur RPC ne nécessite qu'une surcharge de quelques lignes de la méthode DispatchSpecCommand(). (Instructions dans le *guide utilisateur en annexe*). La principale faiblesse réside dans la nécessité de traiter manuellement la sérialisation XML des appels RPC. Toutefois, l'automatisation du marshalling/unMarshalling aurait nécessité un voyage vers les classes d'introspection Delphi dont la durée aurait probablement dépassé le temps prévu pour le projet.

A noter que Windows Vista permet désormais l'interruption d'IO synchrones via de nouvelles API CancelSynchronousIO(). Cette API annule les IO en cours pour un thread donné. Toutefois, le programme d'installation doit fonctionner sur des versions précédentes de Windows où les IO synchrones sont susceptibles de bloquer les threads clients ou serveur.

Mise au point

La principale difficulté de mise au point des classes RPC était due à l'asynchronisme et aux interactions inter-thread. L'identification des sources de blocages à nécessité la mise en place d'un système de traces, intégré au code source de l'unité des classes RPC et activable par définition conditionnelle. Ce code de débogage, placé à certains endroits clés du code, envoient des messages Windows à un programme console et permettent la détection de comportements anormaux ou/et l'origine des blocages.

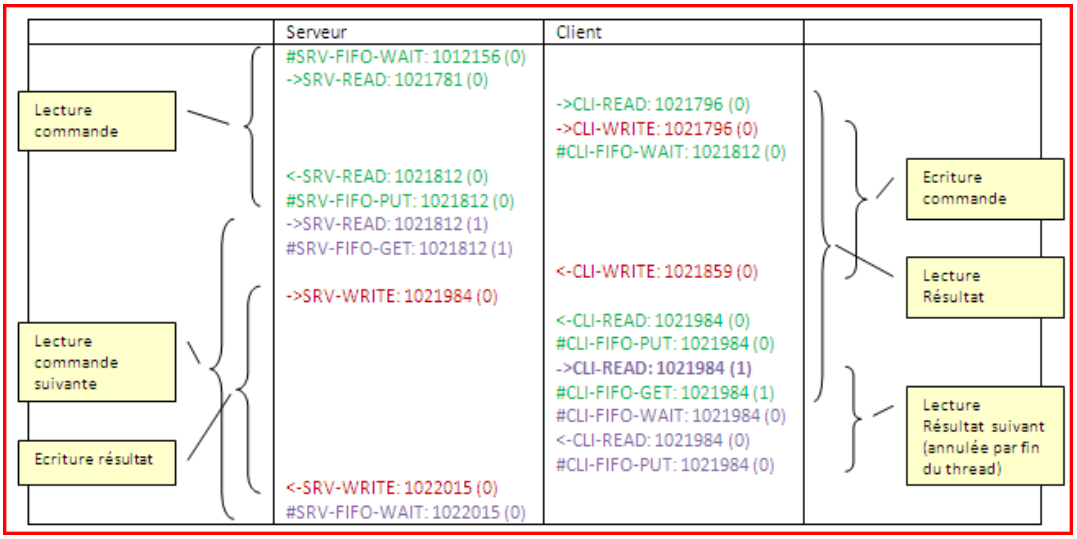


Figure 28 : Exemple de trace d'exécution multi-thread interprétée

Le guide d'implémentation des proxys clients et Stub Serveurs à partir des classes RPC se trouve dans le « manuel utilisateur » en annexe

Service d'installation avec élévation de privilège

Comme mentionné dans le cahier des charges, le programme d'installation à porter sous Windows Vista doit être à même d'exécuter des processus interactifs dans la session utilisateur, sous un compte administrateur local et bien sûr avec un jeton complet.
Comme nous l'avons vu, l'UAC de Windows Vista demande un consentement interactif pour l'élévation de privilège, c'est-à-dire pour pouvoir utiliser un jeton d'accès complet. Bien sûr, ce consentement peut être désactivé (politique de sécurité locale ou de domaine) mais cette désactivation nécessite un redémarrage et fait perdre les avantages de contrôle fournis par UAC et partiellement ceux d'UIPI.
Mentionnons ici, l'absence de solutions d'élévation de privilèges automatique pour Windows Vista au moment de réaliser ce travail. Les meilleures références sur le sujet sont (36) et (37). Le premier donne des exemples en Delphi pour Vista, mais utilise les API LSA et ne crée pas de jeton élevé. Le second décrit la création de jetons à partir de fonctions Windows non documentées (ZwCreateToken()), non compatibles Vista.
L'utilisation des fonctions LSA est prévue pour le développement de fournisseurs d'authentification (p.ex. GINA sous WindowsXP) et la fonctions ZwCreateToken() ne peut créer que des jetons d'accès locaux. Toutefois ces 2 références ont permis d'expérimenter et mieux comprendre les mécanismes de création de jeton et de processus.

L'incertitude sur la faisabilité d'un « runas » avec élévation automatique sans consentement était importante au début de ce projet.

Les principales difficultés ont été :

- Construction / récupération du jeton lié
- Accès aux Winsta/Desktop interactifs
- Interprétation des erreurs de création de processus
- Débogage du service

Le mécanisme implémenté est représentés dans la figure ci-dessous. La solution est satisfaisante dans le sens ou le code reste maintenable (utilisation de fonctions standards et compatible avec les recommandations fournies dans MSDN) et que le résultat correspond à celui attendu. Le processus créé est interactif et son jeton est un jeton administrateur complet.
Lors d'une ouverture de session Windows « normale » (ou avec l'outil ligne de commande Runas.exe de Windows) LSA assigne automatiquement au compte utilisateur, ***un groupe volatile*** de type SE_GROUP_LOGON_ID, identifiant la session interactive. Ce groupe possède les accès aux objets Winsta0 et au Desktop « Default » de la session.
Il n'a pas été possible au cours de ce travail d'assigner ce groupe au jeton créé. La nouvelle API Vista LogonUserExExW() possède un paramètre optionnel pTokenGroups prévu pour ajouter des groupes au nouveau jeton créé, mais elle ne semble pas fonctionner comme décrit dans MSDN.
C'est donc le processus client TL_2.exe qui définit les autorisations sur « ses » Winsta et Desktop, avant d'invoquer le service SIGINST_2.exe. Ce mécanisme peut être jugé limite car il n'est pas certain que les prochaines versions de Windows autorisent un processus non privilégié (compte membre du groupe local Utilisateurs) à modifier les ACL de ces objets (d'autant plus qu'ils sont marqués par des labels d'intégrité HIGH qui devraient, à priori, empêcher la définition des ACL par un processus de niveau MEDIUM !)

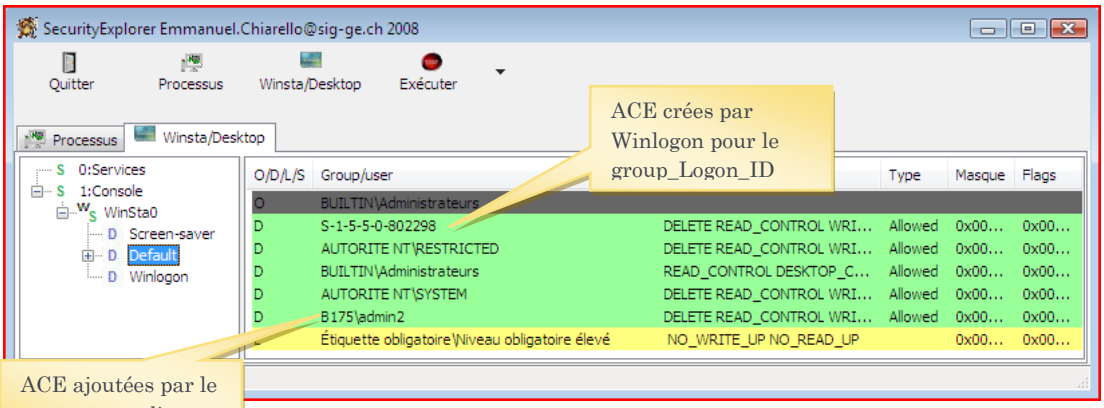


Figure 29 : ACE ajoutées pour autoriser le compte « Admin2 » à accéder au desktop

Pour contourner ce problème, on peut imaginer que le service d'installation exécute tout d'abord un programme non interactif dans la session utilisateur, sous le compte SYSTEM pour définir les permissions nécessaires sur les objets Winsta et Desktop.

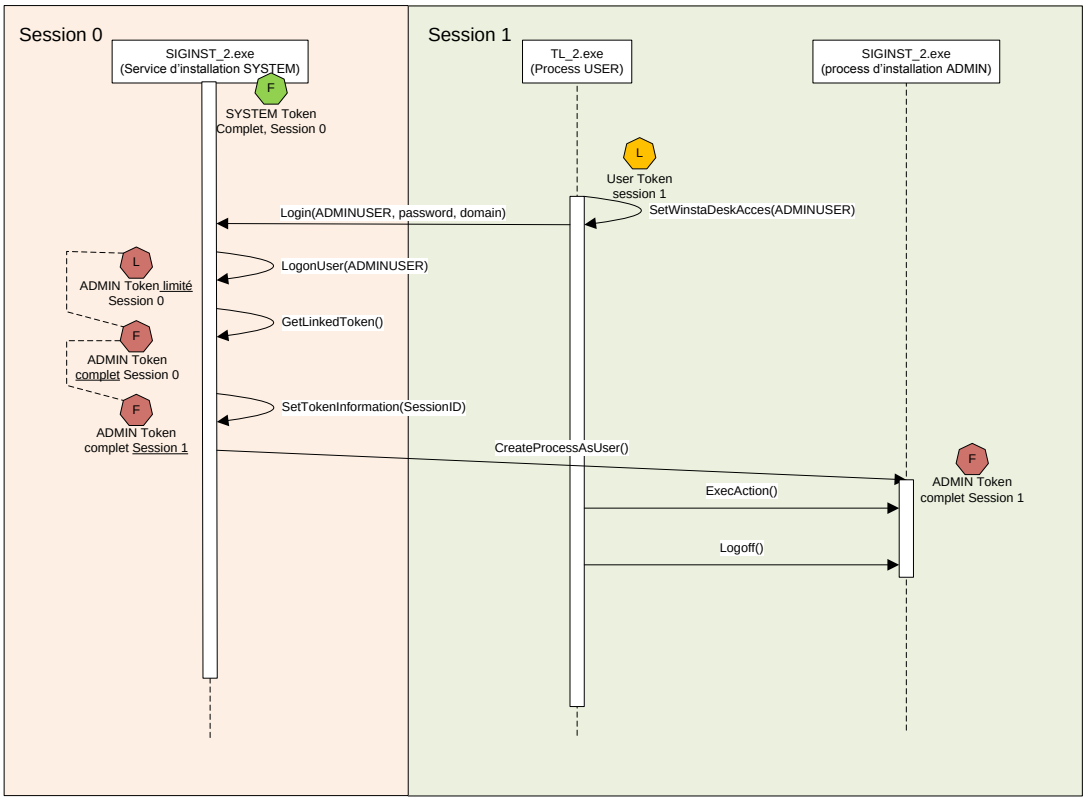


Figure 30 : Diagramme simplifié des interactions du mécanisme d'élévation de privilèges

Accessoirement, le service d'installation peut être utilisé pour démarrer un processus interactif sous le compte SYSTEM dans la session utilisateur. Cette fonctionnalité est très intéressante pour la mise au point des services (particulièrement lors des phases de démarrage, difficiles à tracer avec le débogueur)

Mécanismes de chiffrement

Comme mentionné plus haut, le service d'installation peut être invoqué à distance. Les pipes transmettent les données sur le réseau telles qu'elles.
La procédure de login nécessitant la transmission du mot de passe du compte d'installation (compte de domaine, administrateur local), un mécanisme de chiffrement des données traversant le pipe est nécessaire. Dans ce cas, un « simple » chiffrement à clé symétrique aurait pu être suffisant, avec le risque d'exposer la clé dans le code du programme.

Pour ce travail il m'a donc semblé plus adéquat d'implémenter un mécanisme complet, du type de celui implémenté dans SSL, à l'aide des Crypto API Microsoft (38):

- Négociation des algorithmes de chiffrement optimaux entre le client et le serveur
- Génération d'un couple de clés privée/publique par le serveur et envoi de la clé publique au client
- Génération d'une clé de session symétrique par le client et envoi au serveur, chiffrée avec la clé publique du serveur.

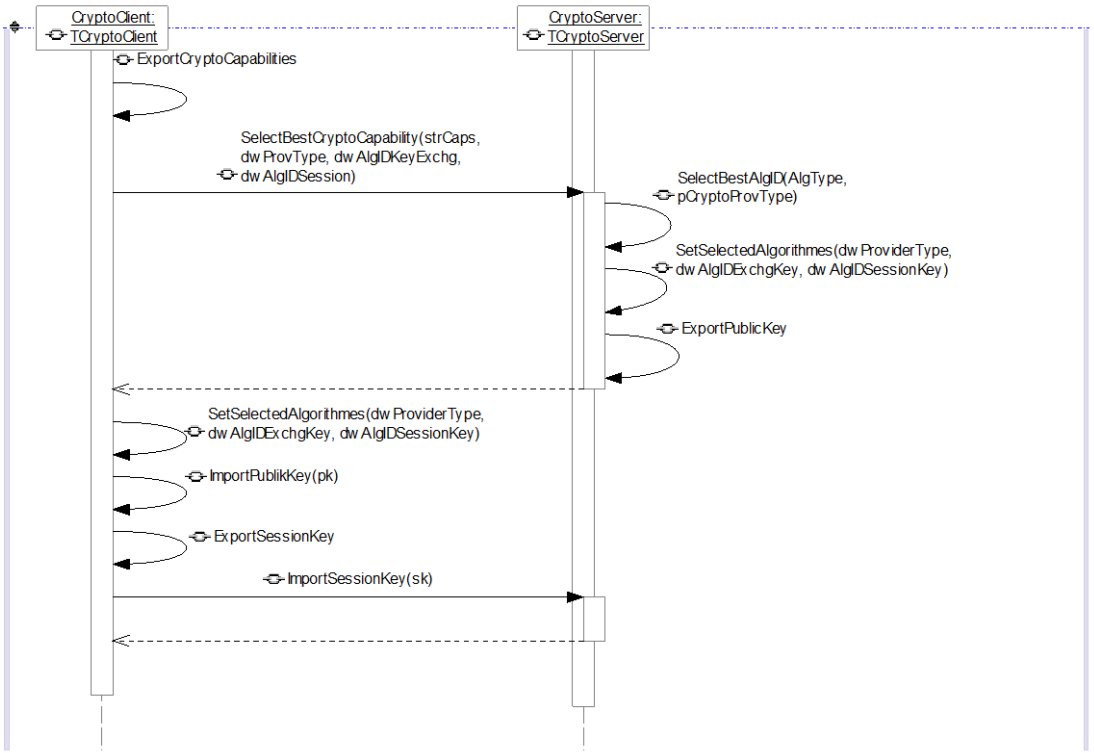


Figure 31 : Diagramme de séquences pour l'établissement du canal sécurisé

L'étude des Crypto API Microsoft a révélé une relative facilité d'utilisation. La mise au point des classes TCryptoClient et TCryptoServer a toutefois nécessité l'intégration d'un code source externe, pour l'encodage/décodage Base64 (39). En effet, le transfert des clés entre le client et le serveur dans les messages XML nécessite un encodage texte, du type Base64, (encodage de $4\times6bits$ ($2^6 = 64$) vers $4\times8bits$ représentant un caractère imprimable). Crypto API propose des API d'encodage/décodage Base64, mais elles ne sont disponibles qu'à partir de Windows XP. Le support de Windows 2000 est requis par le cahier des charges.

L'intégration au niveau des classes RPC s'est faite par dérivation des classes TXMLRPCClient et TXMLRPCServer en TXMLRPCCryptoClient et TXMLRPCCryptoServer et l'ajout d'une interface IStreamInterceptor au niveau de la classe de base TNPStream pour permettre le chiffrement/déchiffrement des données.

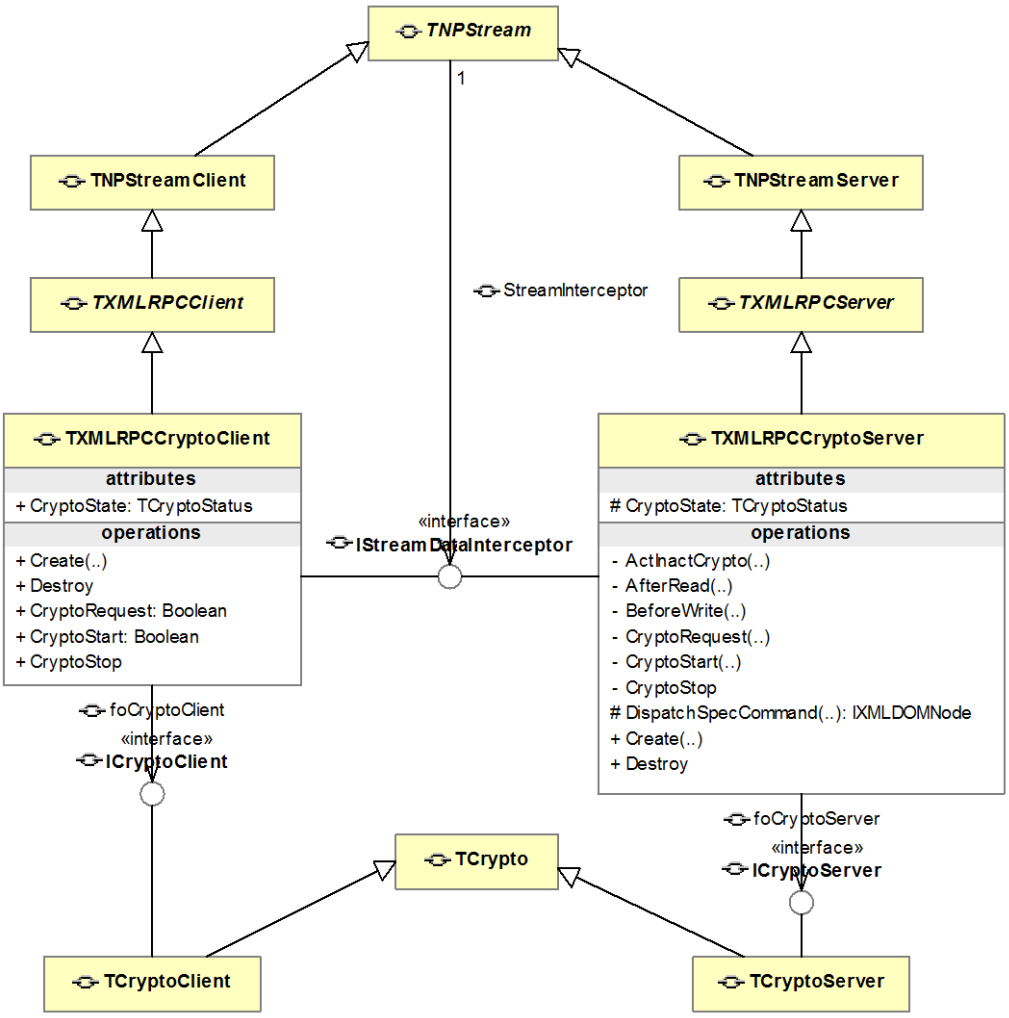


Figure 32 : Intégration des mécanismes Crypto aux classes RPC existantes

Intégration au programme de télédistribution

La modification du programme de télédistribution a été assez délicate à cause du manque de tests unitaires pour le code ancien (parfois plus de 10 ans). L'enjeu étant de garantir l'iso-fonctionnalité tout en augmentant si possible la qualité globale du code existant. L'intégration du service d'installation dans le code du programme client de télédistribution a toutefois été possible avec des modifications bien contrôlées.

Les principales modifications étant:

- Au niveau des actions, remplacement des mécanismes de notification SendMessage() (Information et progression d'installation) par les mécanismes RPC et l'interface IActionNotification.
- Construction et intégration des nouvelles fabriques d'actions génériques en remplacement du code conditionnel existant.
- Remplacement des classes d'action génériques par l'interface IAction.

QUALITE

Tests unitaires

Les principales classes de base ont été systématiquement testées dans un projet de tests unitaires. Ces tests ne couvrent pas toutes les propriétés et méthodes implémentées dans les classes, soit en raison de la trivialité de tests qui n'apporteraient rien, ou alors en raison de difficulté/impossibilité à mettre en place des tests efficaces (le code du test doit rester à un niveau de complexité bien inférieur à celui du code testé).
Par contre, Delphi ne gérant pas la durée de vie des objets (sauf pour les objets des classe TComponent et TInterfacedObject), il est intéressant de pouvoir détecter les éventuelles fuites de mémoire respectivement de handles Windows.
J'ai donc développé une classe TTestCaseExt qui étend la classe TTestCase Delphi. Cette nouvelle classe contrôle l'état de la mémoire et des handles avant et après le test (méthodes setup() et tearDown()). Ces tests ont été d'une grande utilité pour détecter les erreurs de construction dans les structures try finally, chargées du relâchement des handles et zones mémoires alloués.
Toutefois, les mécanismes du runtime et du gestionnaire de mémoire Delphi (garbage collector), peuvent induire des retards dans le relâchement effectif des handles et de zones mémoire. Il est donc parfois nécessaire de définir des « delta » acceptables et/ou de rejouer plusieurs fois les tests. Le problème est particulièrement marqué lors de l'utilisation des objets COM MSXML.
Les mécanismes RPC ont nécessité le développement de programmes auxiliaires propres aux tests. Ces programmes jouent le rôle tantôt de client, tantôt de serveur. Ils sont compilés en même temps que le programme de test et permettent de s'affranchir de certains effets de bord dus à la double synchronisation (côté client et côté serveur) avec le thread principal.

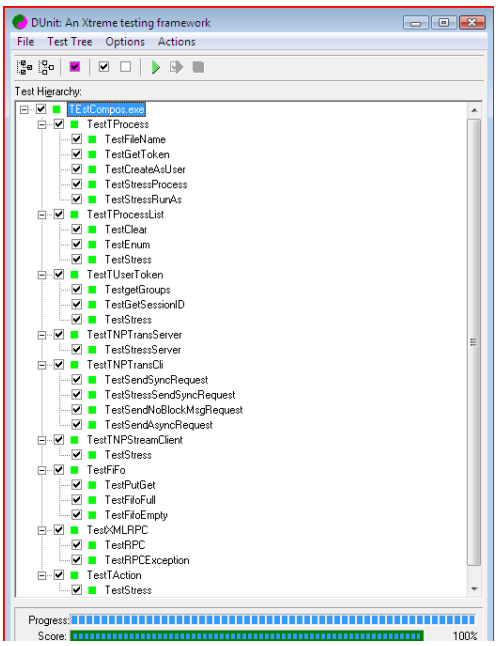


Figure 33 : Résultat de l'exécution des tests unitaires

Ces tests ont été utilisés à plusieurs reprises lors du développement pour valider les non régressions après modification/correction.

Analyse des risques

Risques de sécurité

Le service d'installation mis au point pourrait être considéré comme un point faible dans la sécurité Vista puisqu'il offre un point d'entrée pour le contournement des mécanismes de contrôle d'élévation de privilèges de Vista. C'est pourquoi la sécurisation du service a fait l'objet d'une attention particulière.

Les principales « vulnérabilités » du service sont listées ci-dessous, avec les mesures mises en place pour en limiter l'exploitation ou leurs conséquences :

« Vulnérabilité »	Explication
Mesure (hardening)	
Accès au pipe de communication.	Le service crée 2 pipes de communication sur lesquels les processus clients peuvent se connecter, éventuellement à distance.
Définition des ACL dans le descripteur de sécurité à la création du Pipe. Par défaut, l'accès au pipe est autorisé pour : • L'utilisateur interactif • Le groupe local administrateur (seul ce groupe peut se connecter à distance) • Le compte créateur du pipe (SYSTEM pour le service d'installation et membre du groupe Administrateurs pour le processus d'installation)	
Accès à la méthode de création de processus sous le compte SYSTEM	Le service d'installation supporte la méthode RunAS() qui, sans mentionner de compte utilisateur, exécute un processus sous le compte du service. C'est-à-dire le compte SYSTEM. Sans contrôle, cette méthode permettrait à un utilisateur sans privilège, d'exécuter un processus interactif sous le compte SYSTEM extrêmement privilégié.
Le service teste une ACL spéciale à l'exécution de cette méthode, avec le jeton du client du pipe. Le jeton du client est récupéré par « impersonalisation » du thread et testé par l'API Windows AccessCheck(). Seuls les membres du groupe local Administrateurs y ont accès.	
Privilèges du compte exécutant le service	Le service d'installation doit être exécuté sous le compte SYSTEM en raison de privilèges mis à disposition uniquement de ce compte, tel que « Agir en tant que partie du système d'exploitation » ou « Assigner des jetons d'accès »
Le service est configuré avec ChangeServiceConfig2W() pour être démarré sous le nombre minimum de privilèges requis (les mêmes que le service AppInfo de Vista) soit : • SeAssignPrimaryTokenPrivilege • SeIncreaseQuotaPrivilege • SeTcbPrivilege • SeBackupPrivilege • SeRestorePrivilege • SeDebugPrivilege • SeAuditPrivilege • SeChangeNotifyPrivilege • SeImpersonatePrivilege	

Requête malformée	Une requête malformée envoyée au service peut avoir comme conséquences un dépassement de buffer et pourquoi pas une exploitation de ce dépassement pour exécuter du code arbitraire.
Le thread de lecture du pipe prend l'identité du client à la première lecture de données, limitant ainsi les actions potentiellement néfastes, à ce qu'autorise le compte du client. Le thread de validation XML et d'exécution des actions est exécuté sous un jeton SYSTEM restreint, dont tous les privilèges ont été retirés et dont les groupes de sécurité ont été marqués pour refus seulement. Le message XML est ensuite analysé d'après le schéma XSD embarqué dans l'exécutable du service, sous forme de ressource.	
Exécution de RevertToself() par Buffer overflow	Les mesures de prévention par « impersonalisation » des threads serveurs peuvent être contournées dans le cas d'un buffer overflow, en exécutant l'API revertToSelf() qui a pour effet que le thread revienne au jeton d'exécution de son processus. C'est-à-dire le compte SYSTEM.
Pour limiter ce risque, le service active DEP (prévention d'exécution dans les zones de mémoires de données) avec l'API SetProcessDEPPolicy(), disponible depuis Vista SP1	
Transmission des logon et mot de passe par le réseau	Les mécanismes RPC par named pipe invoqués à distance, transmettent, par défaut, les informations en clair, exposant ainsi les données sensibles comme les mots de passe administrateurs à une éventuelle interception.
Le risque est limité par l'implémentation des mécanismes de chiffrement. Clé asymétrique pour le chiffrement et la transmission sûre de la clé symétrique de session. La négociation préalable des paramètres cryptographiques assure que les meilleurs algorithmes communs au client et au serveur sont utilisés.	

Analyse du code

L'analyse des métriques du code a été effectuée à l'aide du programme **CodeHealer** (40). Ce programme fournit des métriques, résultant de l'analyse du code Delphi, dont la plus significative pour une analyse de risques est la **« complexité cyclomatique »**, fonction du nombre de branchements (if-else, try-except) et de boucles (for, while, …). Cette valeur reflète plus ou moins la complexité structurelle du programme et donc le risque lié à la qualité et à la « maintenabilité » du code.

Complexité structurelle	Evaluation du risque pour un module ou un programme
1-10	Relativement simple, sans grand risque
11-20	Plus complexe, risque modéré
21-50	Complexe, risque élevé
+ 50	Non testable, risque très élevé

Tableau 6 : Evaluation du risque en fonction de la complexité cyclomatique (40)

Programme/module	Complexité moyenne	Unités utilisées (non uniques)	Nb. de lignes
SIGINST_2.EXE	4	719	53'410
TL_2.EXE	4	749	53'622
ACT32_2.DLL	4	666	50'434
PROCEXPLORER.EXE	4	148	16'947
Acl2.pas	4	10	2'893
Uact.pas	4	46	1'142
Uactremote.pas	3	13	762
Uactplus.pas	5	19	599
Ucrypto.pas	6	5	1'209
Udeskwinsta.pas	2	4	717
Unpstream.pas	4	17	2'713
Uprocess2.pas	3	16	1'097
Usiginstallsvc.pas	3	27	927
Utoken.pas	4	7	754

Tableau 7 : Métriques des principaux programmes et modules du projet

On constate que les unités développées au cours de ce projet ont une complexité structurelle faible (2 à 6) et donc un risque lié faible. Toutefois, CodeHealer a permis de mettre en évidence 3 unités préexistantes dont la complexité structurelle et donc le risque lié est plus élevé. Ces unités devront être revues afin d'en diminuer le risque.

UActCEReg.pas =21, UActReg.pas =16, UActIcon.pas =12

A noter que certaines unités ayant peu de lignes et des structures de type if… else if … else if … même très simples, peuvent fournir une complexité cyclomatique élevée pas forcément représentative du niveau de risque réel.

Intégration

Les nouvelles versions des composants du programme « SIGTL » ne sont pas compatibles avec ceux utilisés jusqu'ici. De plus, s'agissant du programme chargé des mises à jour, la mise à jour de celui-ci doit se faire avec prudence.
Le déploiement devrait se faire en 2 phases :

- Installation des nouveaux composants et du nouveau service, par le « SIGTL » existant
- Désinstallation des anciens composants par le nouveau « SIGTL »

Il y aura toutefois une période transitoire où certains ordinateurs seront équipés de l'ancienne version du programme « SIGTL » et d'autres de la nouvelle version. C'est pourquoi les nouveaux composants n'ont pas le même nom que leurs prédécesseurs :

Description	Ancien nom	Nouveau om
Programme client d'installation	TL32.EXE	TL_2.exe
Service d'installation	SIGInstall.exe	SIGINST_2.exe
Librairie des actions d'installation	PIBACT32.dll	ACT32_2.dll

Tableau 8 : Correspondance des noms des anciens et des nouveaux composans « SIGTL »

Il faudra prévoir un mécanisme capable de choisir le bon programme client d'installation en fonction de ce qui est installé (et fonctionnel). La détection du nouveau service d'installation peut se faire en testant l'existence des pipes du service.

D'autre part, les fichiers exécutables seront marqués avec un fichier manifest embarqué, indiquant le niveau d'exécution et éventuellement la librairie des contrôles graphiques à utiliser.

Finalement, ils seront signés (technologie Authenticode Microsoft) avec un certificat interne à SIG (voir guide utilisateur en annexe D).

Fichier Manifest

Un fichier « Manifest » est un fichier au format XML qui accompagne le fichier exécutable, soit comme fichier compagnon (nomDuProgramme,exe.manifest) soit comme ressource embarquée (voir *Guide utilisateur* en annexe).
L'intérêt du fichier manifest dans le cadre du déploiement d'applications est de contrôler le niveau d'exécution requis par un programme. A noter que la présence d'un fichier manifest désactive la redirection des dossiers et clés de registre du système de virtualisation de Vista.

Marquage	Description	Virtualisation (redirection)
Pas de manifest	Appel comme AsInvoker	Oui
AsInvoker	L'application s'exécute avec le même jeton d'accès que le parent	Non
HighestAvailable	L'application s'exécute avec les privilèges les plus hauts que l'utilisateur puisse obtenir	Non
RequireAdministrator	L'application ne peut-être exécutée que par un administrateur et requiert que l'application soit démarrée avec le jeton complet d'un administrateur.	Non

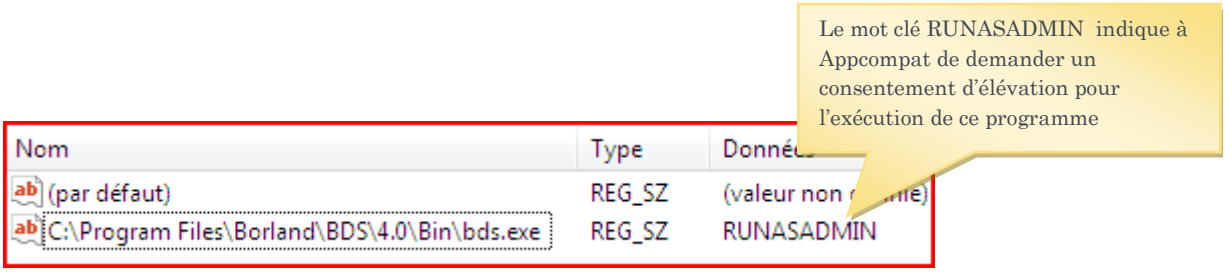
Tableau 9: Valeurs pour le niveau d'exécution requis dans un manifest (27)

Une autre manière de contrôler le niveau d'exécution d'un programme est d'inscrire une valeur spéciale au niveau du registre Windows :

```
HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers
```

Ou

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers
```



Conclusion

Ce travail a été l'occasion de mener à terme un projet complexe et enrichissant, par les sujets abordés comme par les méthodes et la diversité des techniques utilisées. L'expérience retirée de cette étude restera certainement valable sur le long terme.

J'ai également pu vérifier le potentiel et les limites du développement orienté modèle pour Delphi. La mise en pratique des méthodes d'analyse et de développement (UML et tests unitaires) a été une expérience enrichissante, difficilement accessible dans mon domaine professionnel axé sur l'intégration et l'exploitation des infrastructures informatiques.

Pour SIG qui a financé cette formation MAS-ICT, le retour sur investissement est concrètement réalisé par la disponibilité d'un outil de télédistribution répondant à un besoin réel, actuellement en phase de **validation**.

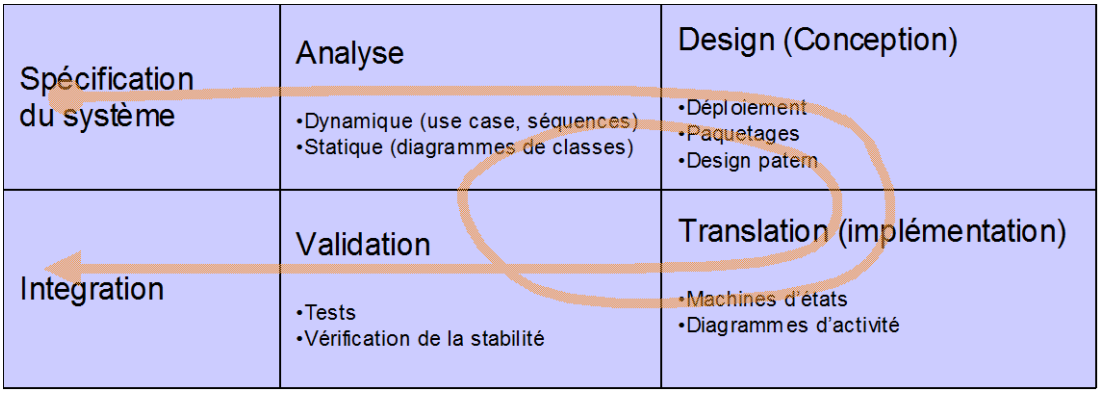


Figure 34 : Modèle 6q HEVs

La suite à donner à ce projet dépendra des décisions de la hiérarchie informatique SIG, mais les tests fonctionnels approfondis (comme l'installation et la gestion complète d'un PC sous Windows Vista) pourraient avoir lieu courant décembre 2008 et aboutir en janvier 2009 à une procédure de déploiement de Windows Vista sur les ordinateurs portables SIG (dans un premier temps).

Emmanuel Chiarello 27 novembre 2008

Bibliographie

1. **Disson, Éric (Uni Genève).** *Ingénierie des modèles d'accès :.* 14 11 2007.

2. Bell-Lapadula. *Wikipedia.* [En ligne] http://en.wikipedia.org/wiki/Bell-LaPadula_Model.

3. Biba model. *Wikipedia.* [En ligne] http://en.wikipedia.org/wiki/Biba_model.

4. DOD TCSEC. [En ligne] http://csrc.nist.gov/publications/history/dod85.pdf.

5. Contrôle d'accès discrétionnaire. *Wikipedia.* [En ligne] http://fr.wikipedia.org/wiki/Contr%C3%B4le_d%27acc%C3%A8s_discr%C3%A9tionnaire.

6. Contrôle d'accès obligatoire. *Wikipedia.* [En ligne] http://fr.wikipedia.org/wiki/Contr%C3%B4le_d%27acc%C3%A8s_obligatoire.

7. Trusted computing base. *Wikipedia.* [En ligne] 06 2008. http://en.wikipedia.org/wiki/Trusted_computing_base.

8. Microkernel. *Wikipedia.* [En ligne] 06 2008. http://en.wikipedia.org/wiki/Microkernel.

9. *Trusted System Concepts.* **Abrams, M.D, Michael, ph.D et Joyce, V.** Oxford : The MITRE Corporation, 1995. Computers & Security N° 1.

10. Outils pour le chiffrement des données pour PC mobiles - Analyse de la sécurité. *www.Microsoft.com.* [En ligne] 04 04 2007. [Citation : 01 12 2008.] http://www.microsoft.com/france/technet/security/guidance/clientsecurity/dataencryption/analysis/4e6ce820-fcac-495a-9f23-73d65d846638.mspx.

11. Winobjs. *Sysinternals.* [En ligne] 06 2008. http://technet.microsoft.com/fr-fr/sysinternals/bb896657(en-us).aspx.

12. **Russinovich, Mark.** *Windows Vista Kernel Changes.* Vista kernel change by M. Russinovitch.pptx.

13. **Poublan, Jean-Yves.** *Renforcement et sécurité des services Windows Vista/Longhorn.* Jour1-251-4-Renforcement_Services_Vista.pptx.

14. About Atom Tables. *msdn.* [En ligne] http://msdn.microsoft.com/en-us/library/ms649053(vs.85).aspx.

15. **Howard, Michael et LeBlanc, David.** *Writing secure code for Windows Vista.* [éd.] Microsoft. Secure software Development series. s.l. : Microsoft, 2007. Vol. 1. ISBN-13: 978-0-7356-2393-4.

16. Shatter attack. *Wikipedia.* [En ligne] [Citation : 01 10 2008.] http://en.wikipedia.org/wiki/Shatter_attack.

17. **Associates, Maze &.** Windows Vista Security. [En ligne] [Citation : 01 10 2008.] http://www.learnsecurity.org/Shared%20Documents/Vista-security-seminar.pdf.

18. **Ourghanlian, Bernard.** *Fonctionnement interne de Windows Vista.* s.l. : Microsoft, 2007.

19. **MSDN, Microsoft.** What is the Windows Integrity Mechanism. *msdn.microsoft.com.* [En ligne] [Citation : 15 09 2008.] http://msdn.microsoft.com/en-us/library/bb625957.aspx.

20. **Riley, Steve.** Steve Riley on Security. *blogs.technet.com.* [En ligne] 5 11 2007. [Citation : 28 10 2008.] http://blogs.technet.com/steriley/archive/2006/07/21/Mandatory-integrity-control-in-Windows-Vista.aspx#455338.

21. **Microsoft.** Windows Integrity Mechanism Design . *MSDN.* [En ligne] [Citation : 01 09 2008.] http://msdn.microsoft.com/en-us/library/bb625963.aspx.

22. **Poublan, Jean-Yves.** *User Account Control.* s.l. : Microsoft France. TechDays 2007.

23. **Greene, Thomas C.** Win32 API utterly and irredeemably broken. *biznix.org.* [En ligne] [Citation : 01 10 2008.] http://www.biznix.org/articles/25883.html.

24. **Barbosa, Edgar.** Windows Vista UIPI. *coseinc.com.* [En ligne] [Citation : 01 10 2008.] http://www.coseinc.com/Vista_UIPI.ppt.pdf.

25. Client/Server Runtime Subsystem. *Wikipedia.* [En ligne] [Citation : 01 10 2008.] http://en.wikipedia.org/wiki/CSRSS.

26. **Rutkowska, Joanna.** The official blog of the invisiblethings.org. *theinvisiblethings.blogspot.com.* [En ligne] [Citation : 01 10 2008.] http://theinvisiblethings.blogspot.com/2007/02/running-vista-every-day.html.

27. *Migration d'une application vers Windows Vista.* **d'application., Jacques Massa - Consultant Senior en développement.** 02 2007, Programmez!, p. 27.

28. **Microsoft.** *Windows Vista Application Development Requirements for User Account Control Compatibility.*

29. —. Windows Vista Software Logo Spec 1.1. *Microsoft.com.* [En ligne] http://download.microsoft.com/download/8/e/4/8e4c929d-679a-4238-8c21-2dcc8ed1f35c/Windows%20Vista%20Software%20Logo%20Spec%201.1.doc.

30. **Russinovich, Mark.** Inside Windows Vista User Account Control. *Technet.* [En ligne] [Citation : 01 11 2008.] http://technet.microsoft.com/en-us/magazine/cc138019.aspx.

31. **SysInternals/Microsoft Marc Russinovich.** Process explorer v11.21. *www.sysinternals.com.* [En ligne] [Citation : 1 6 2008.] http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx.

32. Shell code generator. *metasploit.* [En ligne] [Citation : 01 10 2008.] http://metasploit.com:55555/PAYLOADS.

33. **Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides.** Patrones de diseño (notación UML). *www.vico.org.* [En ligne] [Citation : 01 07 2008.] http://www.vico.org/pages/PatronsDisseny.html.

34. DCOM Architecture. *MSDN.* [En ligne] [Citation : 01 11 2008.] http://msdn.microsoft.com/en-us/library/ms809311.aspx.

35. **Microsoft.** [MS-SSTP]: Secure Socket Tunneling Protocol (SSTP) Specification. *MSDN.* [En ligne] [Citation : 01 10 2008.] http://msdn.microsoft.com/en-us/library/cc247338.aspx.

36. **FaNIX.** Starting a Interactive Process in Vista using TService. *delphipraxis.net.* [En ligne] http://www.delphipraxis.net/topic120296_starting+a+interactive+process+in+vista+using+tservice.html&highlight=createprocessasuser.

37. Zhefu Zhang. *codeproject.com.* [En ligne] 24 10 2006. http://www.codeproject.com/KB/system/RunUser.aspx.

38. Cryptography. *msdn.* [En ligne] [Citation : 01 10 2008.] http://msdn.microsoft.com/en-us/library/aa380255(VS.85).aspx.

39. How to Base 64 (MIME) Encode and Decode a String. *Delphifaq.net.* [En ligne] [Citation : 01 10 2008.] http://www.delphifaq.net/how-to-base-64-mime-encode-and-decode-a-string/.

40. CodeHealer. *www.socksoftware.com.* [En ligne] [Citation : 01 11 2008.] http://www.socksoftware.com/codehealer.php.

41. **CodeGear.** Delphi 2009. *ww.codegear.com.* [En ligne] [Citation : 1 10 2008.] http://www.codegear.com/products/delphi/win32.

42. **modelmakertools.** ModelMaker: Native Refactoring and UML 2.0 modeling for Delphi and C#. *modelmakertools.com.* [En ligne] [Citation : 01 10 2008.] http://www.modelmakertools.com/modelmaker/index.html.

43. **Altova.** XMLSpy. *www.altova.com.* [En ligne] [Citation : 1 10 2008.] http://www.altova.com/products/xmlspy/xml_editor.html.

44. DelphiCodeToDoc. *dephicodetodoc.sourceforge.net.* [En ligne] TridenT. [Citation : 1 10 2008.] http://dephicodetodoc.sourceforge.net/.

45. **Johnson, Angus.** Télécharger Delphi Code Converter . *www.clubic.com.* [En ligne] [Citation : 1 10 2008.] http://www.clubic.com/telecharger-fiche122636-delphi-code-converter.html.

46. **VMWare.** VMWare Workstation. *www.vmware.com.* [En ligne] [Citation : 1 10 2008.] http://www.vmware.com/products/ws/.

47. **Microsoft.** SQL Server Express Edition. *MSDN.* [En ligne] [Citation : 01 10 2008.] http://msdn.microsoft.com/fr-fr/express/aa718378.aspx?wt.srch=1.

48. GUI-Based RunAsEx. *codeguru.* [En ligne] http://www.codeguru.com/cpp/w-p/win32/cursors/article.php/c6745__2/.

49. TOKEN_INFORMATION_CLASS Enumeration. *msdn.* [En ligne] 06 2008. http://msdn.microsoft.com/en-us/library/aa379626(VS.85).aspx.

50. logon session . *msdn.* [En ligne] http://msdn.microsoft.com/en-us/library/ms721592(VS.85).aspx.

Tableaux et illustrations

INDEX

Livrables

Programmes (\Sources\bin.32)	Nom	Statut
Service d'installation SIGTL	SIGINST_2.exe	Remplace
Programme console d'installation client SIGTL	tlcmd_2.exe	Adapté
Librairie d'exécution des actions SIGTL	act32_2.dll	Adapté
Programme GUI d'installation SIGTL	tl_2.exe	Adapté
Programme GUI de test des actions d'installation	Tstact_2.exe	Adapté
Programme de validation des tests unitaires	TEstCompos.exe	Nouveau
Programme de test auxiliaire, utilisé par TestCompos.exe	TestAuxNP.exe	Nouveau
Programme de test auxiliaire, utilisé par TestCompos.exe	TestNPStream.exe	Nouveau
Programme auxiliaire de debug des mécanismes de lecture/écriture des pipes	DebugNP.exe	Nouveau
Utilitaire de test des mécanismes UIPI	testIntegrity.exe	Nouveau
Utilitaire de test du mécanisme DEP	TestDEP.exe	Nouveau
Utilitaire de test des Crypto API	TestCrypto.exe	Nouveau
Utilitaire d'exploration des processus	ProcExplorer.exe	Nouveau
Utilitaire permettant la visualisation des descripteurs de sécurité des objets fichier et registre	ACLDump.exe	Nouveau
Utilitaire client du service d'installation. Permet d'exécuter un programme sous un autre compte administrateur, sans demande de consentement.	Runas.exe	Nouveau

Unités des composants (code\Sources\Compos)	Nom	Statut
Unité de gestion des descripteurs de sécurité des objets	acl2.pas	Adapté
Unité externe : traduction des API BCrypt Microsoft	BCrypt.pas	Non utilisé
Fonctions diverses	fctdiv.pas	Adapté -
Unité externe : traduction des API NCrypt Microsoft	NCrypt.pas	Non utilisé
Unité de gestion des Services Windows	Service.pas	Adapté
Unité des déclarations SIGTL	SiDecl.pas	Adapté -
Unité de gestion des données de connexion des ressources réseau	SiSqlCon.pas	Adapté -
Unité de gestion des réplications de fichiers	SiSqlRep.pas	Adapté -
Unité de gestion des données de télédistribution	SiSQLTL.pas	Adapté +
Unité étendue du service manager Delphi	SvcMgrEx.pas	Etendu +
Unité de base des actions de télédistribution	UAct.pas	Adapté ++
Unité des actions de télédistribution	UAct___.pas	Adaptés -
Unité des éléments d'actions SIGTL étendus	UActPlus.pas	Adapté, nouvelle unité
Unité d'exécution des actions distantes	UActRemote.pas	Nouveau
Unité des objets de cryptographie CryptoAPI	UCrypto.pas	Nouveau
Unité des objets Winsta et Desktop Windows	UDeskWinSta.pas	Nouveau
Unité des classes de communication client serveur et mécanismes RPC sur Named pipe	UNPStream.pas	Nouveau

Unité des classes de communication transactionnelle sur named pipe	UNPTrans.pas	Nouveau
Unité de gestion des processus	UProcess2.pas	Nouveau
Unité contenant les définissions relatives au service d'installation	USigInstallDef.pas	Nouveau
Unité des session « terminal server » Windows	UTermServ.pas	Nouveau
Unité de gestion des jetons d'accès Windows	UToken.pas	Nouveau
Unité de traduction des API LSA Windows	UWinLSA.pas	Nouveau
Unité de traduction des API Userenv Windows	UWinUserenv.pas	Nouveau
Unité de traduction des API Windows Vista et autres, manquant dans Windows.pas de Delphi	UWinVista.pas	Nouveau
Unité de traduction des API CryptoAPI Windows http://cc.codegear.com /Item.aspx ?id=17598)	WinCrypt.pas	Importé

Modèles UML (ModelMaker 9) et schémas XSD (code\Modèles)	**Nom**	**Statut**
Diagramme d'étude des mécanismes RPC	Design construction.mpd	Brouillons de travail
Diagrammes des classes du projet	Modèle final.mpd	Référence
Diagramme des relations entre objets Windows	Relation entre objets Windows.mpb	Documentation
Diagrammes d'activité des interactions entre client et serveur d'un pipe	UML Named-pipes.vsd	Documentation
Diagramme de séquences décrivant le mécanisme d'élévation de privilèges	UML interactions service SIGTL.vsd	Documentation
Schéma XSD des messages XML client-serveur	SIGActionMessages.xsd	Référence

Documentation du projet (code\doc)	**Nom**	**Statut**
Fichier d'aide Windows de documentation des classes Delphi du projt	Sécurité Windows VISTA.chm	Référence
Rapports d'analyse des métriques du code	Metrics/metrics-xxx.pdf	Référence
Description initiale du mandat	cahier des charges Etude UAC Windows.pdf	Historique
Etude préalable, cahier des charges détaillé	00 cahier des charges.pdf	Référence
Mémoire du projet	01 Mémoire.pdf	Référence
Journal du projet	02 Journal.pdf	Historique
Manuel d'utilisation des programmes et classes du projet	03 manuel utilisateur.pdf	Référence

Outils

Cette section liste les différents outils utilisés au long de ce projet, dans l'ordre d'importance décroissant.

- Turbo Delphi 2006 de CodeGear/Borland (41)
- Model maker 9 (42)
- Process Explorer de SysInternals (31)
- WinObjs de SysInternals
- XMLSpy 2006 de Altova (43)
- DelphiCodeToDoc de TridenT (44)
- DelphiCodeConverter de Angus Johnson (45)
- CodeHealer for Delphi 2.5 (40)
- VMWare Workstation 6.03 de VMWare Inc (46)
- SQLServer express 2005 (47)

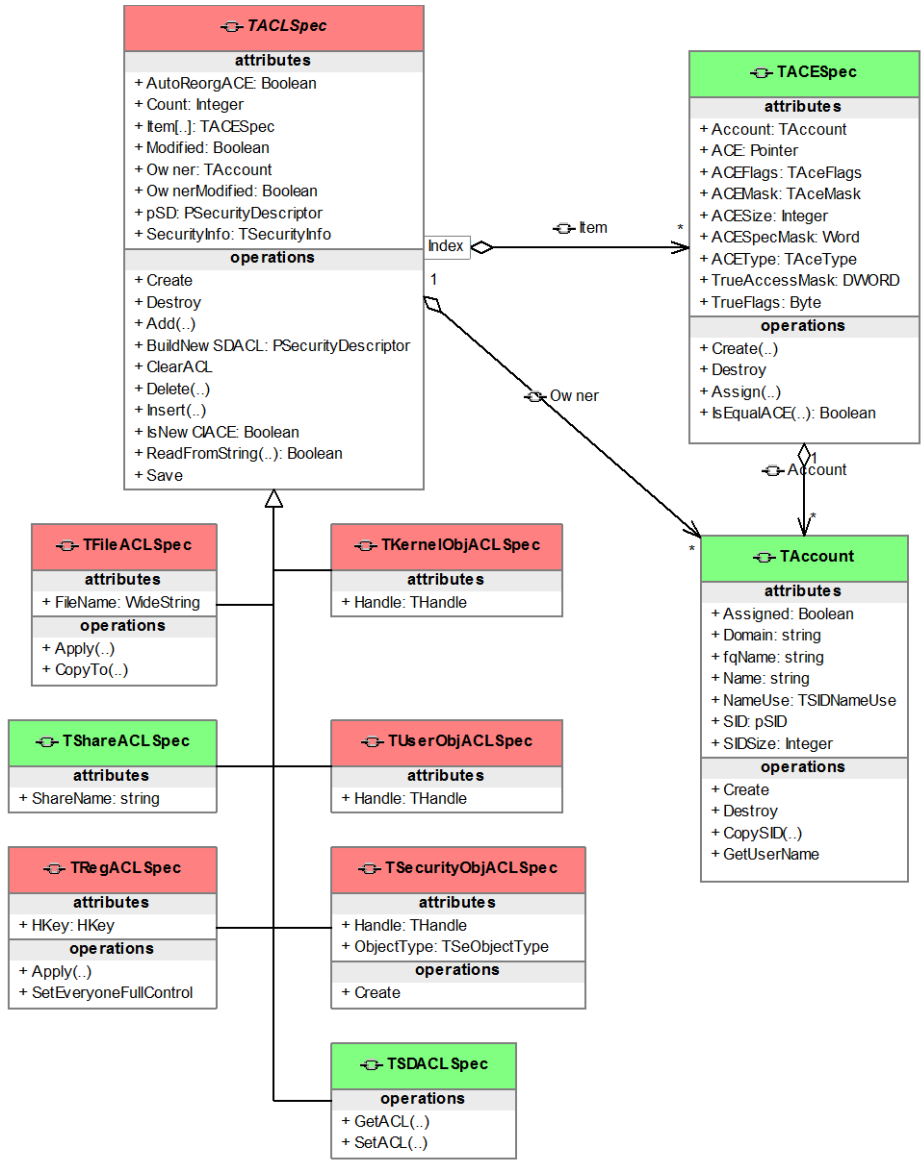
Diagrammes structurels

Les schémas suivants représentent les principales classes développées, modifiées ou simplement utilisées au cours de ce projet.

- Les classes préexistantes, non modifiées mais utilisées,sont représentées en vert
- Les classes existantes modifiées/adaptées sont représentées en rouge
- Les classes nouvellement crées sont représentées en jaune

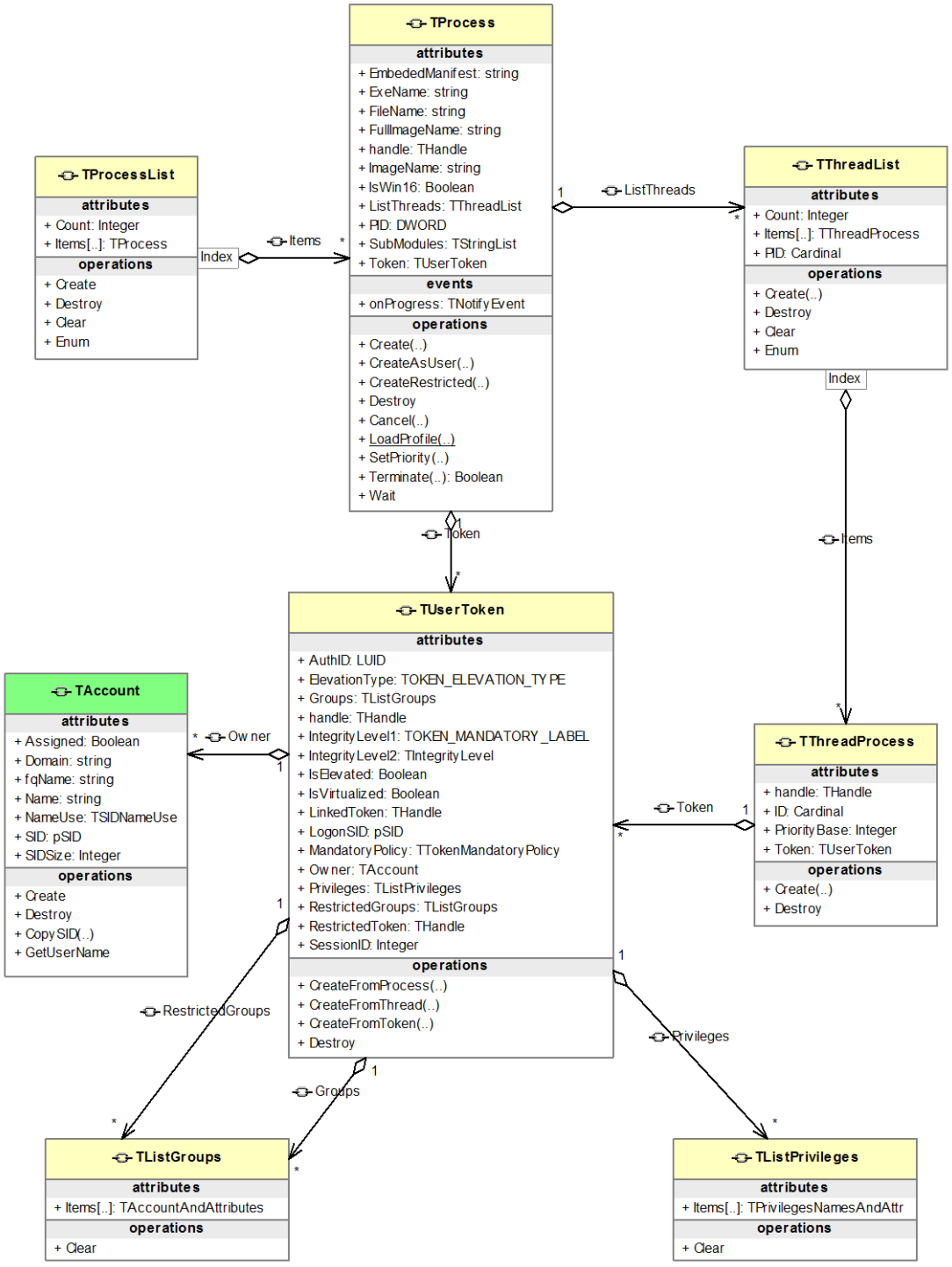
Classes de manipulation des descripteurs de sécurité

Modifiées pour accéder aux SACL et aux labels d'intégrité de Windows VISTA.

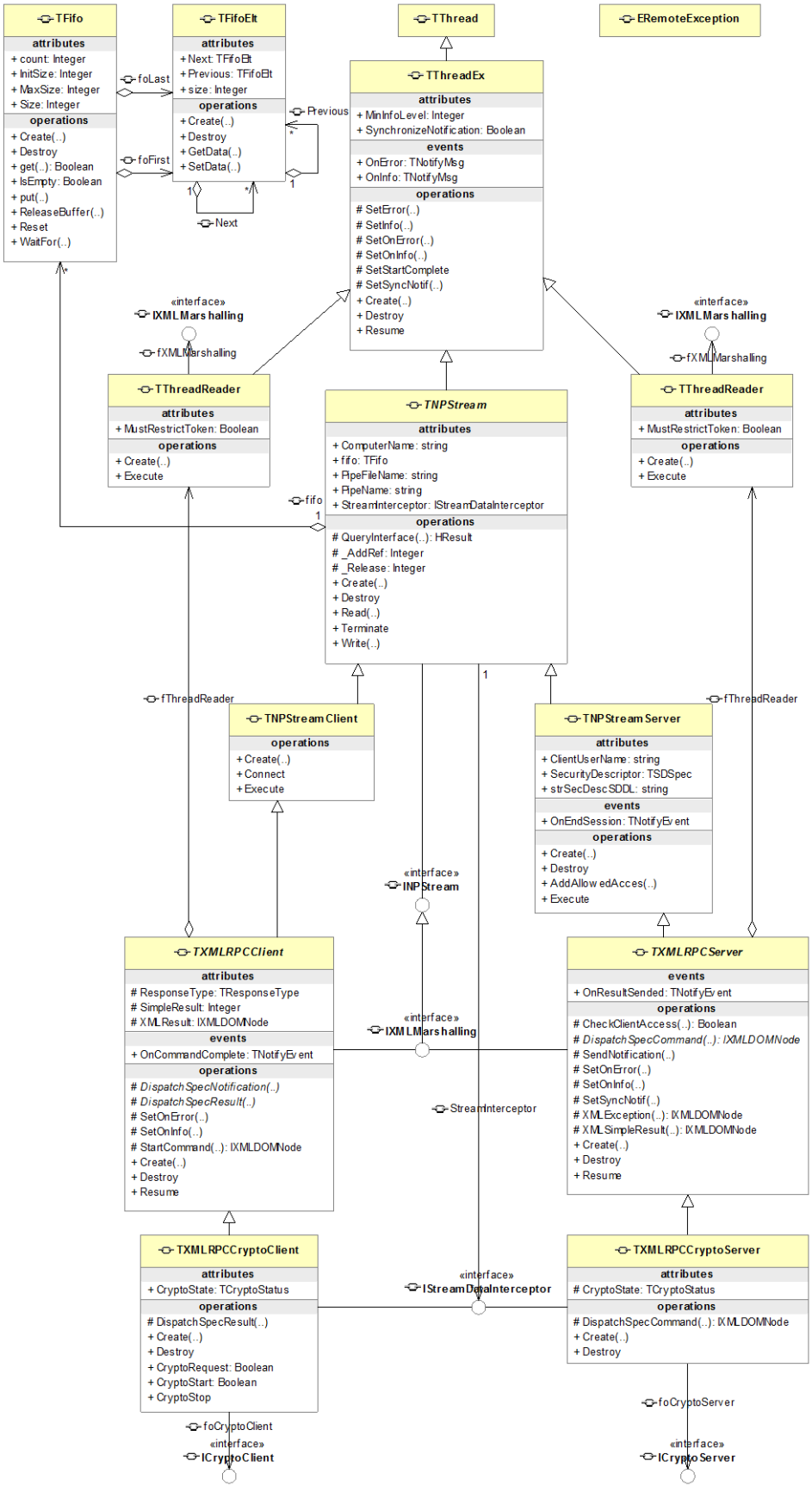


Classes de gestion des processus, jetons utilisateurs et threads.

Ces classes sont nouvelles, sauf la classe TAccount préexistante.

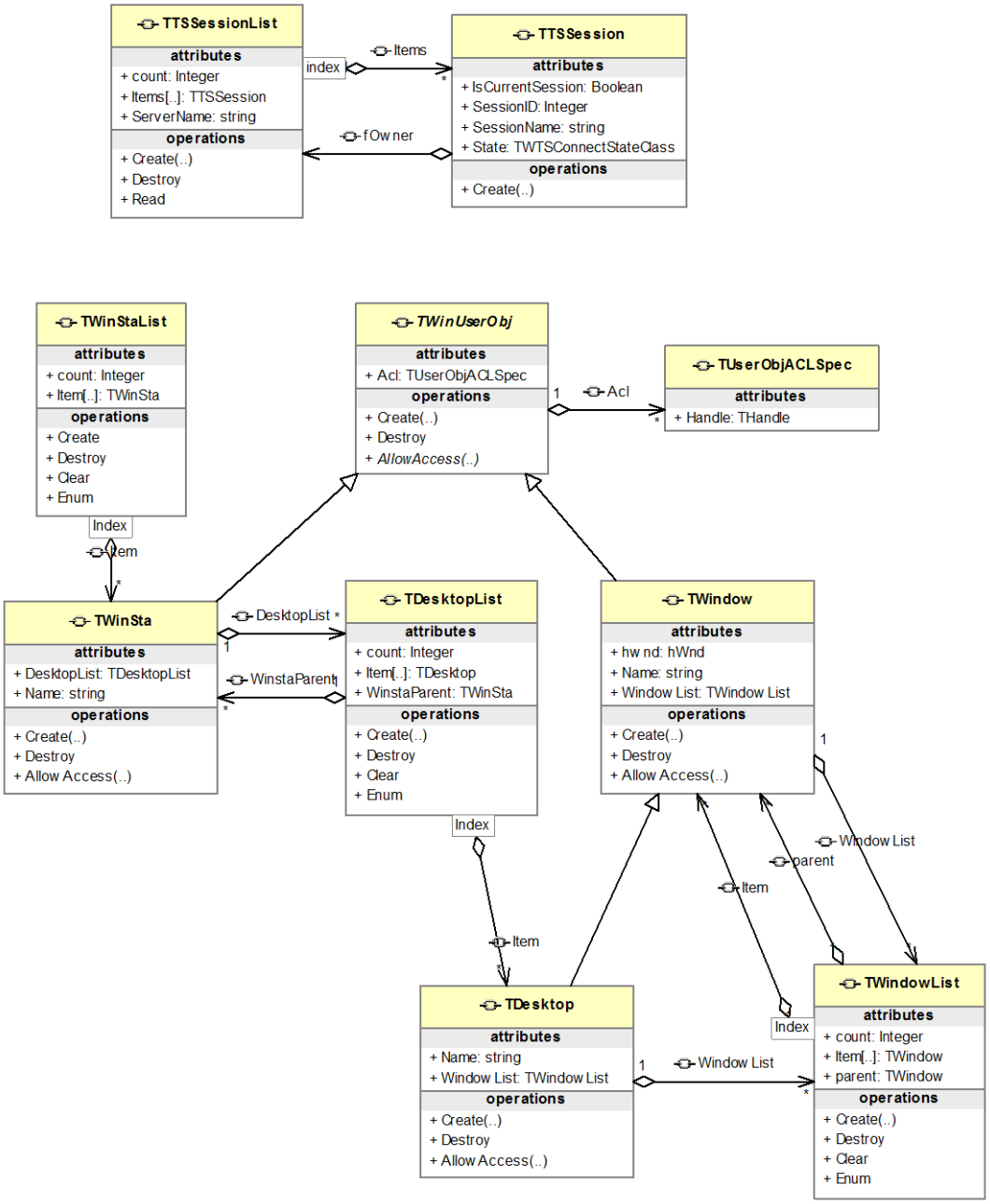


Classes du mécanisme RPC sur named-pipe



Classes de gestion des objets Session, Winsta et Desktop.

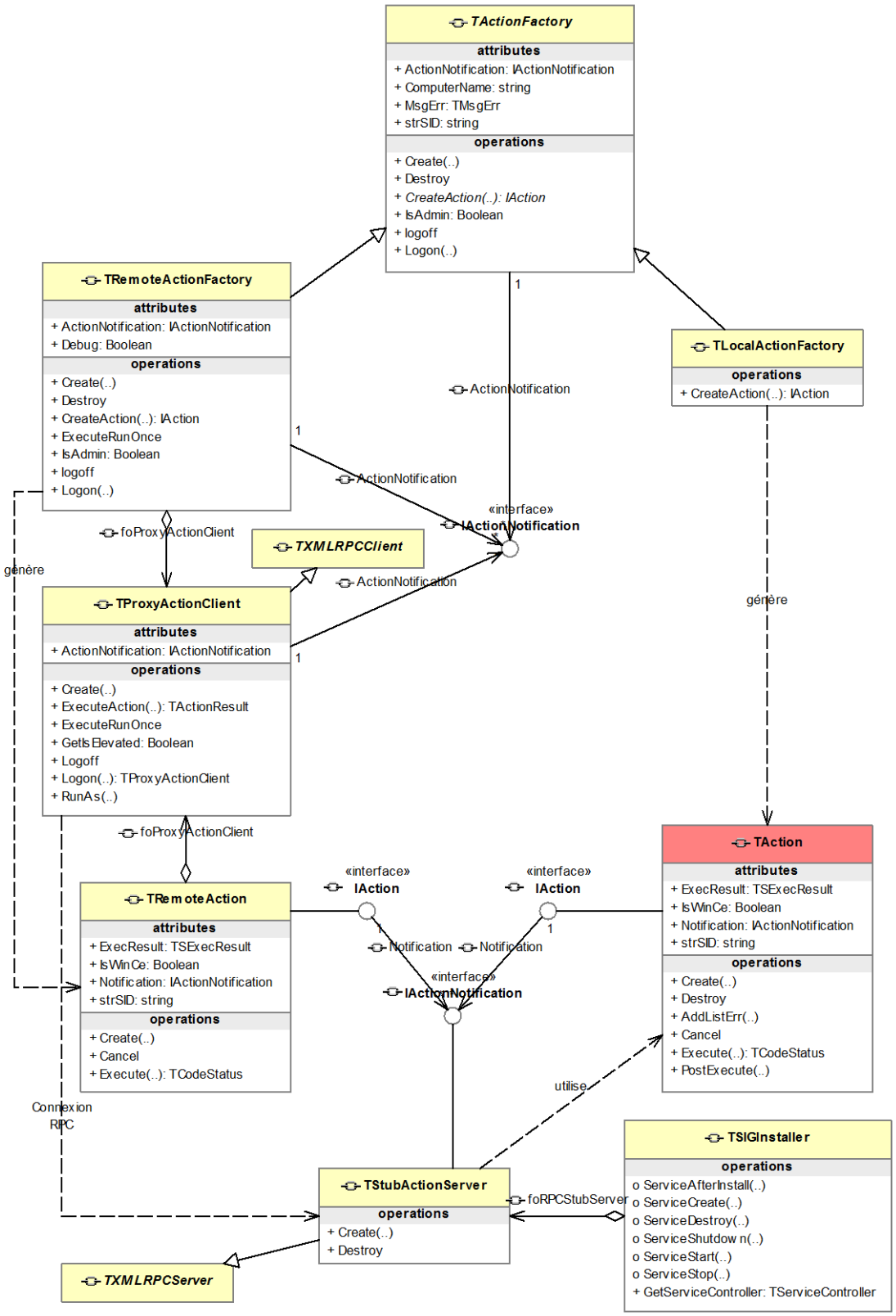
Toutes ces classes sont nouvelles.



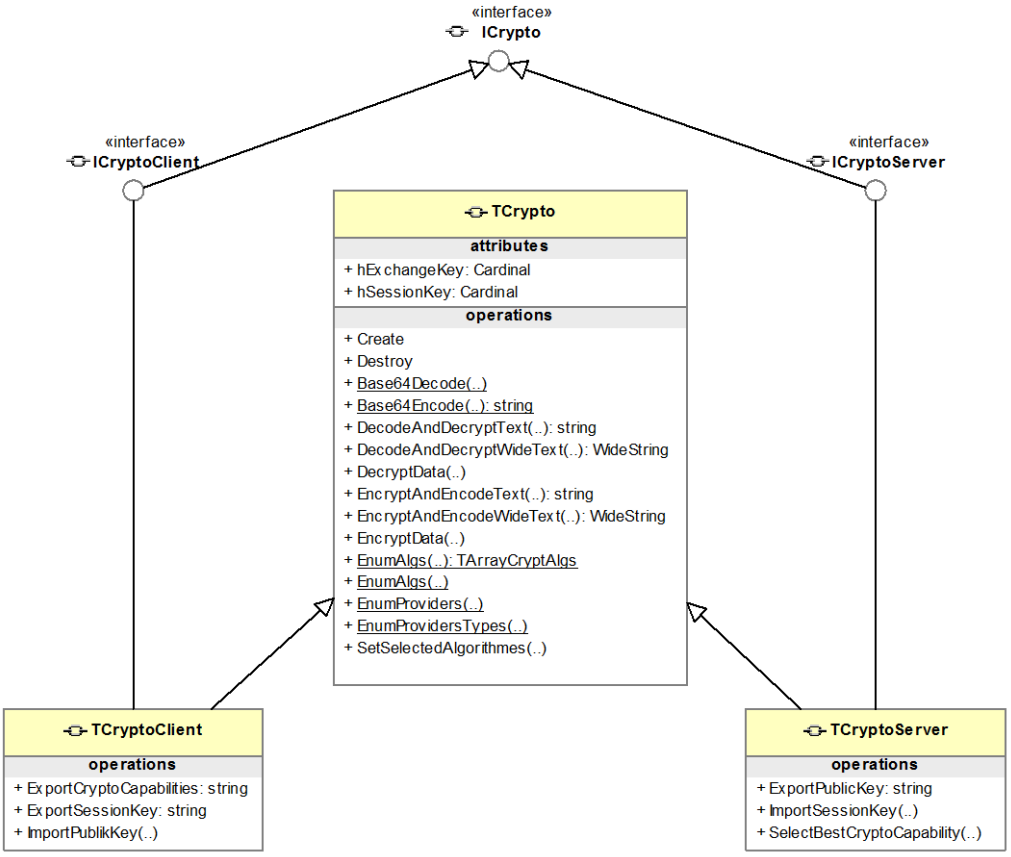
Classes en relation avec les actions de télédistribution

Actions fabriques d'actions, classes de communication RPC et service d'installation.

Toutes ces classes sont nouvelles, sauf la calsse TAction, adaptée pour l'usage des interfaces IAction et IActionNotification.

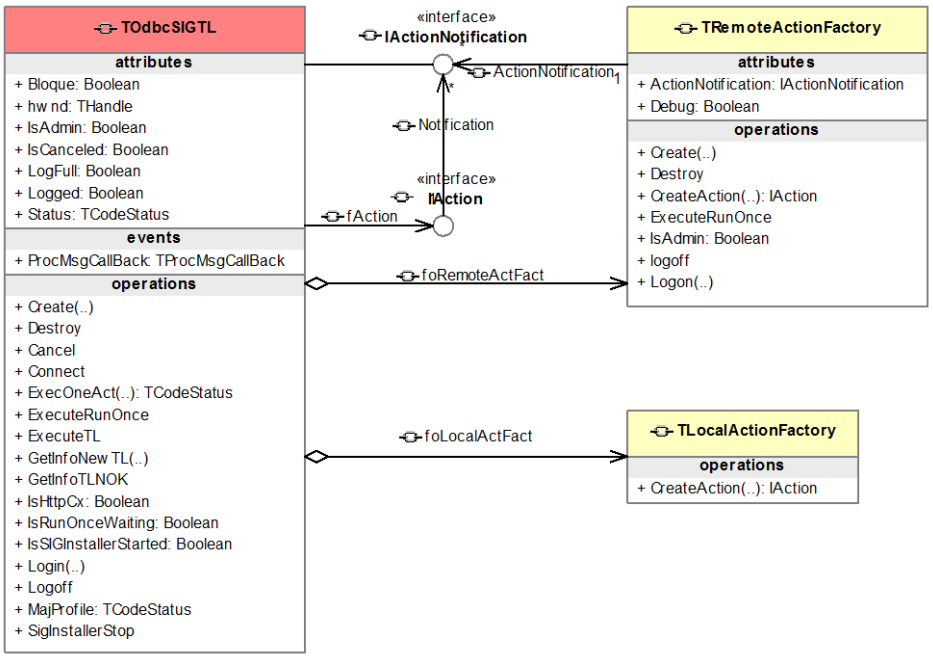


Classes de chiffrement



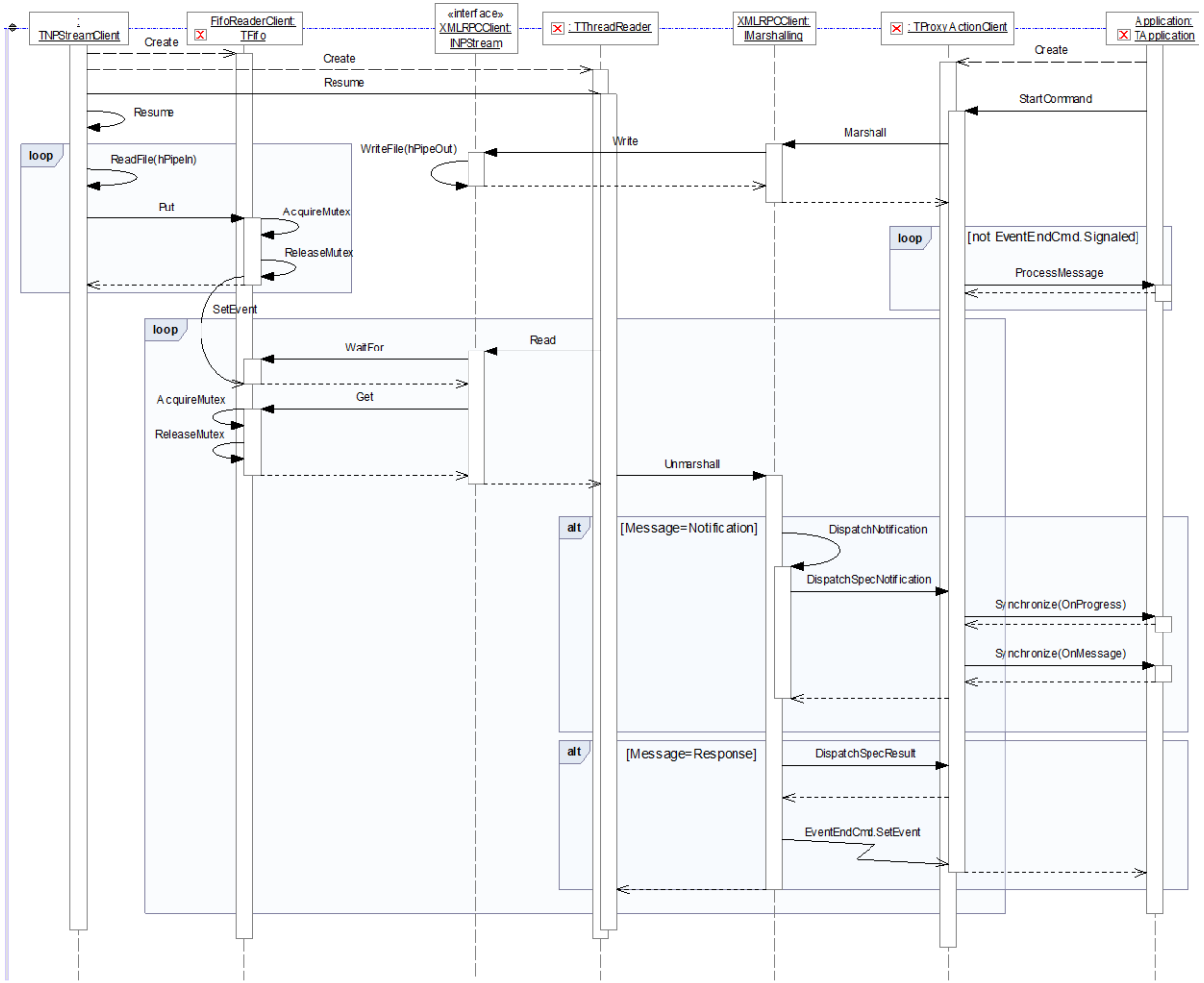
Intégration et fabriques d'actions

Intégration des fabriques d'actions au programme de télédistribution existant (classe TOdbcSIGTL).



Diagrammes dynamiques

Diagramme dynamiques des classes RPC client



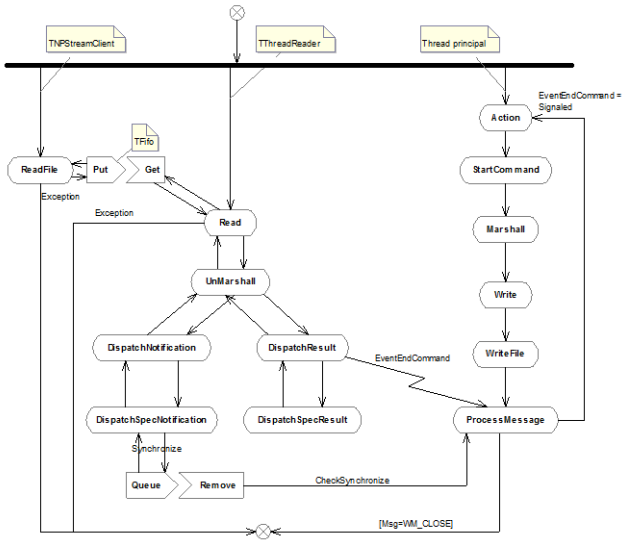


Diagramme dynamiques des classes RPC serveur

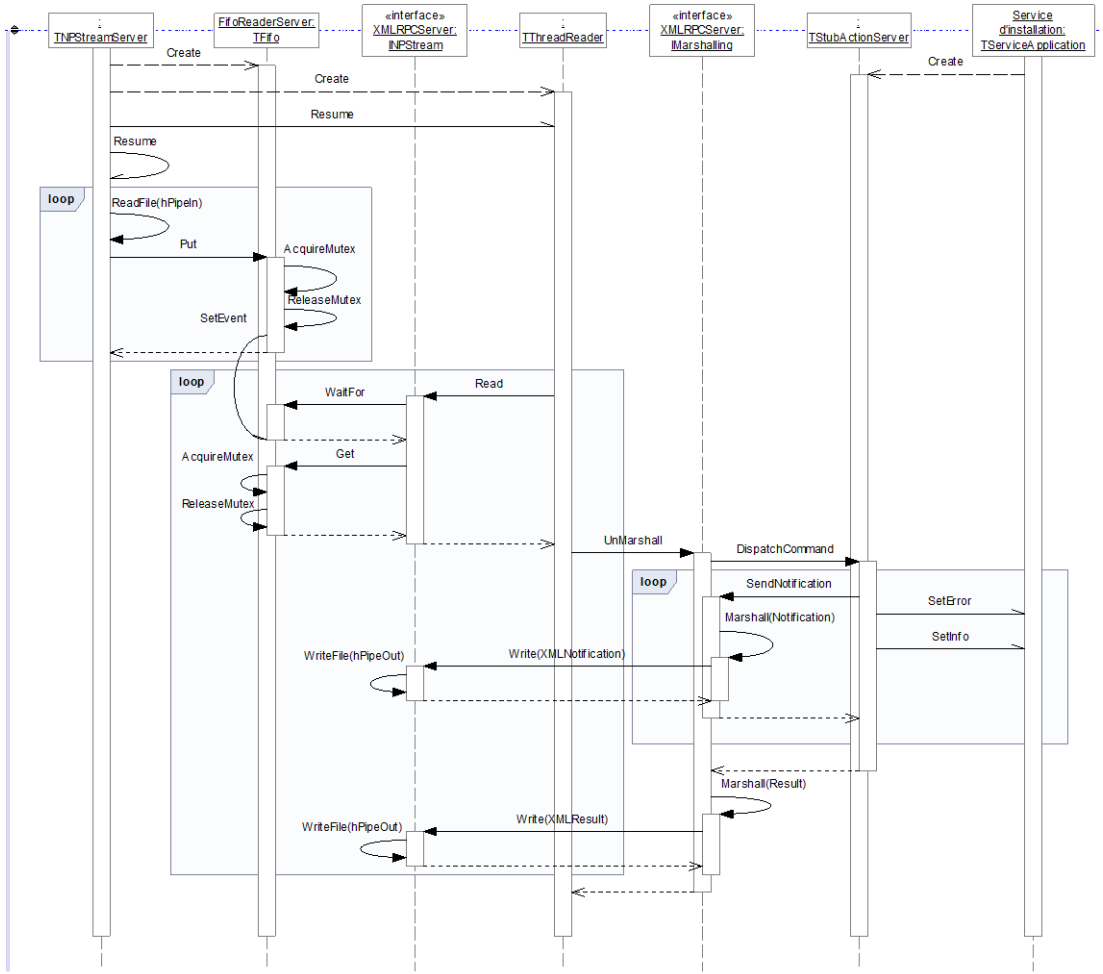
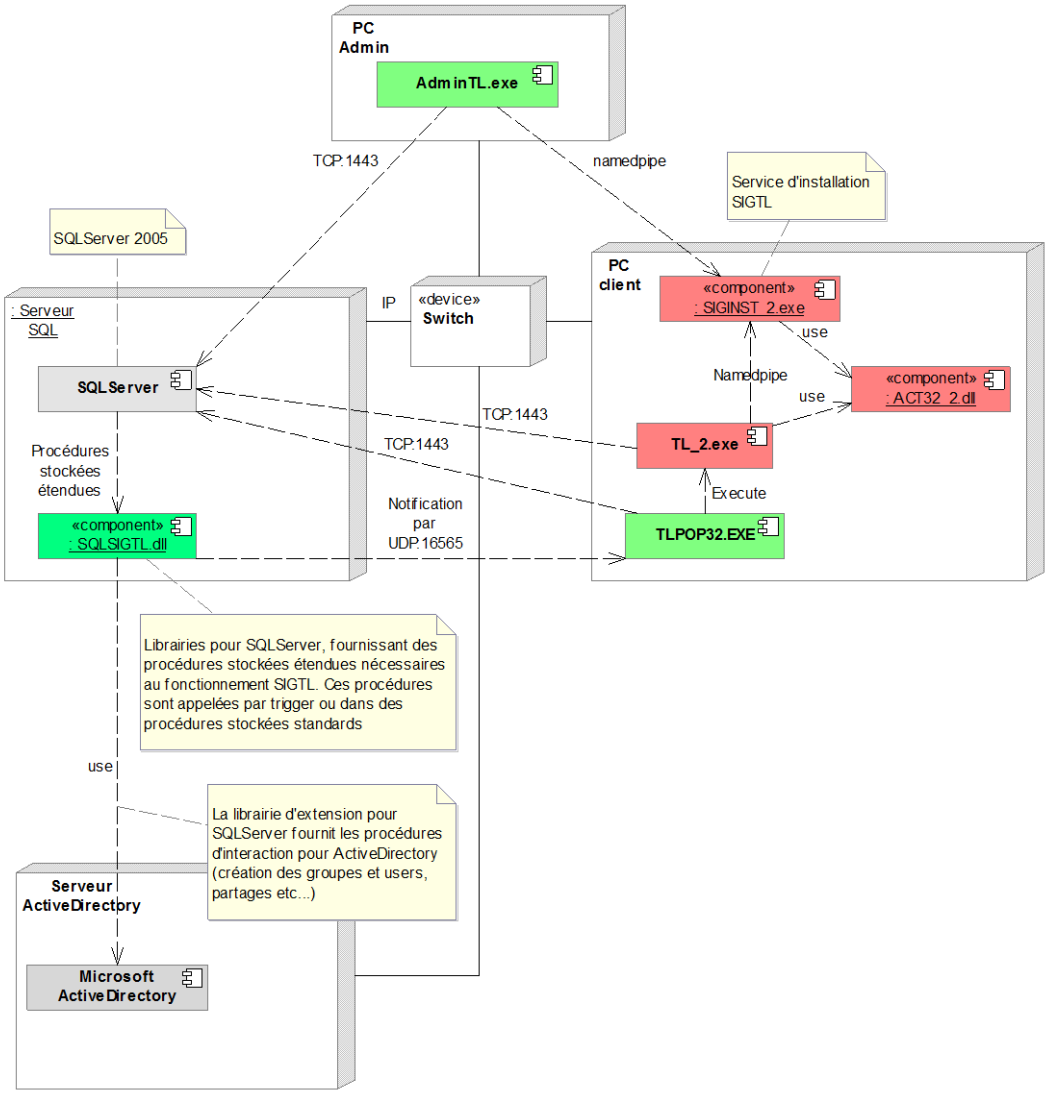


Diagramme d'implémentation



Principales unités et classes du projet

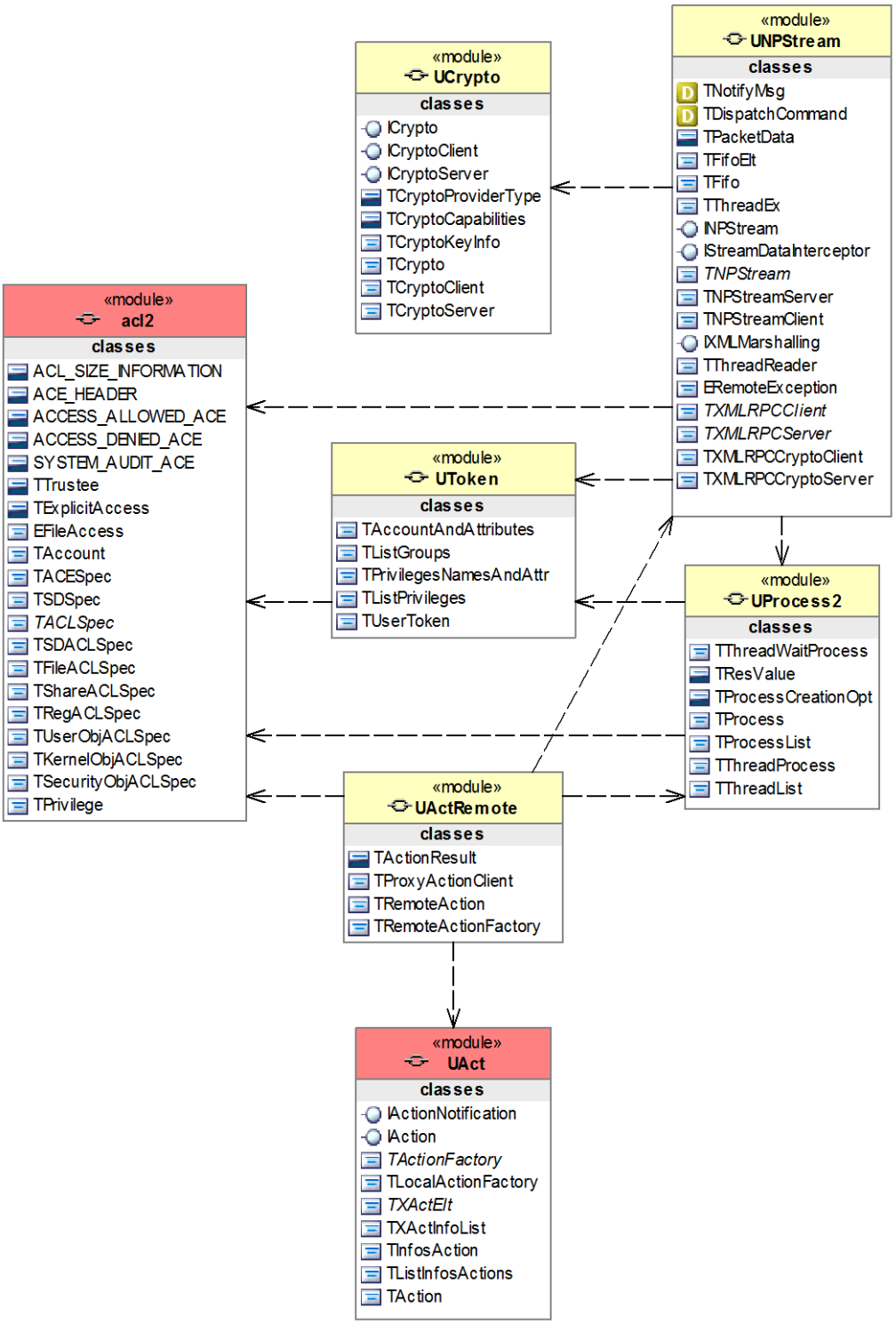
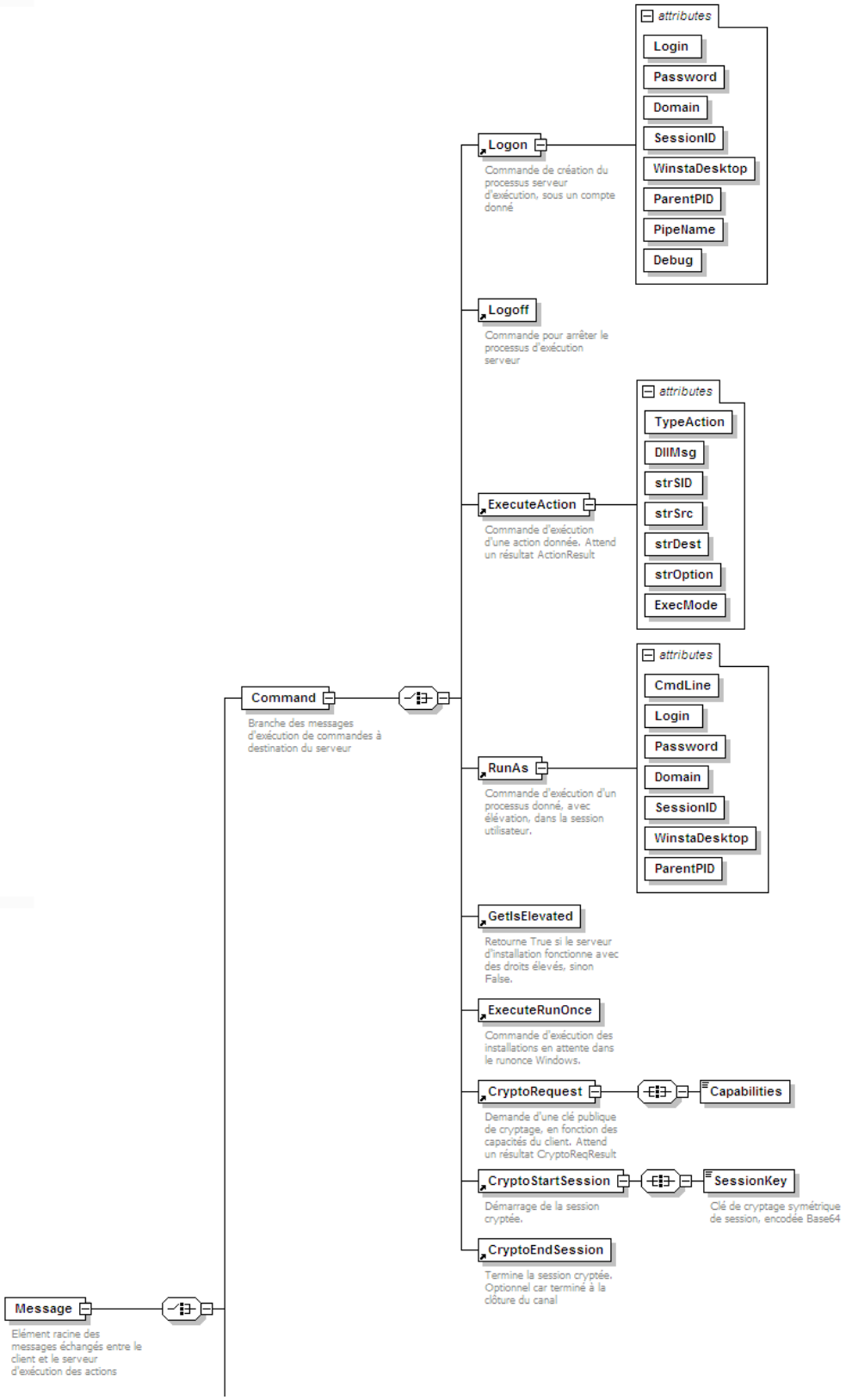
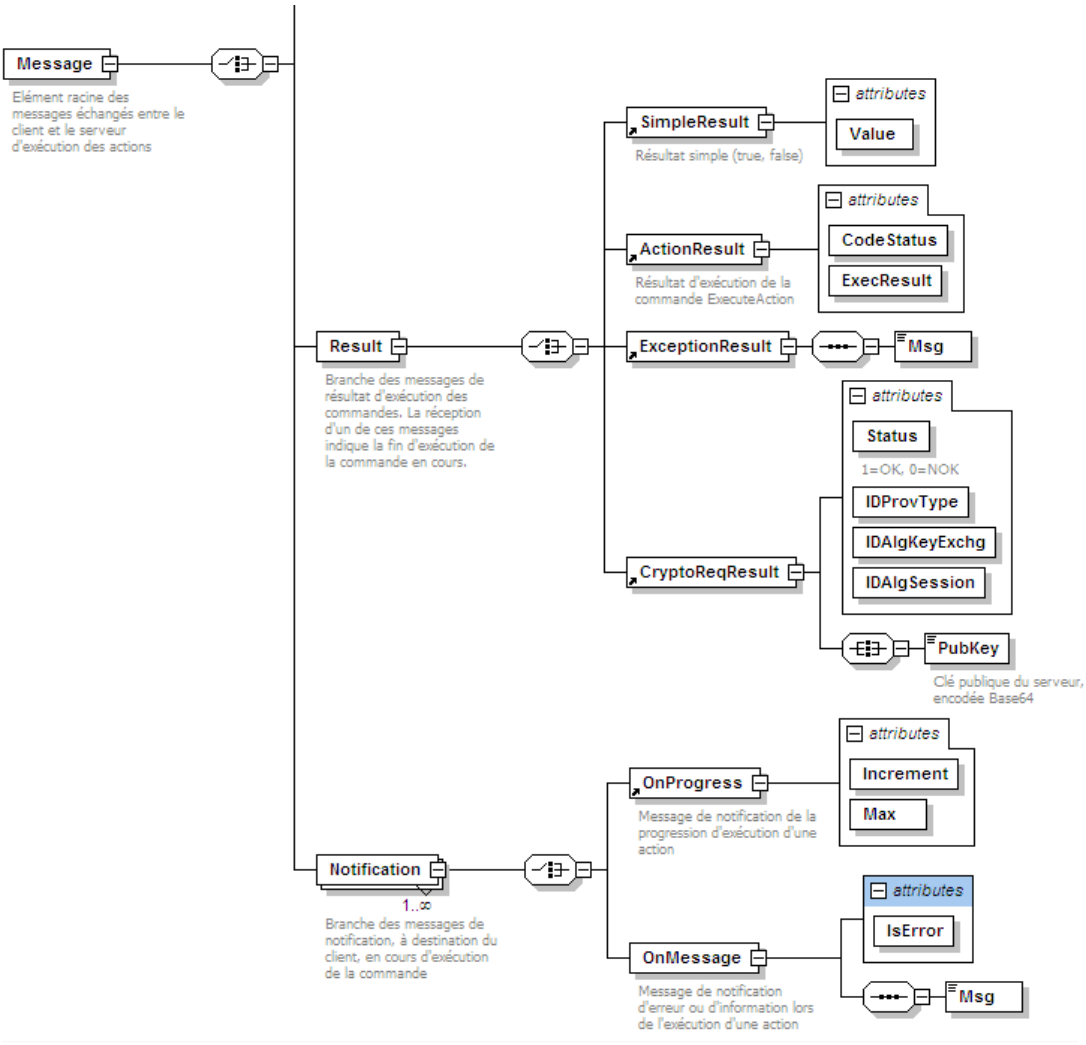


Schéma XML des commandes et réponses RPC

Schéma XSD des messages supportés par le service d'installation.




Message
Elément racine des messages échangés entre le client et le serveur d'exécution des actions
Result
Branche des messages de résultat d'exécution des commandes. La réception d'un de ces messages indique la fin d'exécution de la commande en cours.
SimpleResult
Résultat simple (true, false)
attributes
Value
ActionResult
Résultat d'exécution de la commande ExecuteAction
attributes
CodeStatus
ExecResult
ExceptionResult
Msg
CryptoReqResult
attributes
Status
1=OK, 0=NOK
IDProvType
IDAlgKeyExchg
IDAlgSession
PubKey
Clé publique du serveur, encodée Base64
Notification
1..∞
Branche des messages de notification, à destination du client, en cours d'exécution de la commande
OnProgress
Message de notification de la progression d'exécution d'une action
attributes
Increment
Max
OnMessage
Message de notification d'erreur ou d'information lors de l'exécution d'une action
attributes
IsError
Msg

Annexes

A. Proposition initial de mandat
B. Composants du jeton d'accès Windows Vista
C. Liste des dossiers et clés du registre marquées par un label d'intégrité explicite
D. Manuel utilisateur

Annexe A:

PROPOSITION INITIAL DE MANDAT

Contexte

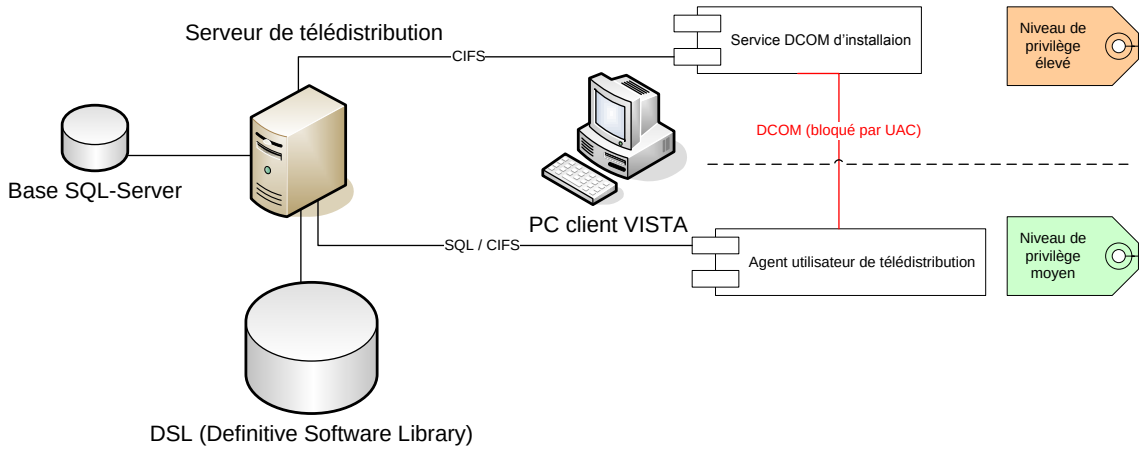
Windows VISTA intègre de nouvelles fonctionnalités pour augmenter la sécurité des postes de travail. L'UAC (User Account Control) en est le principal composant. Si cette fonctionnalité ne pose pas de problèmes particuliers sur un PC privé, ce n'est pas le cas en entreprise.
Cette technologie complexe freine actuellement le déploiement de Windows VISTA en entreprise. Elle doit être comprises et maîtrisées pour permettre un déploiement et un fonctionnement conforme des applications des postes de travail.
Par exemple, le programme de télédistribution de logiciels utilisé et développé à SIG, ne fonctionne pas avec l'UAC activé (problème identifié au niveau de la communication interprocessus DCOM).

Description du mandat

Comprendre en détail les mécanismes de sécurité de Windows VISTA et en particulier UAC.
Identifier les problématiques et les solutions pour le déploiement des applications existantes.
Adapter le programme de télédistribution SIG pour le rendre compatible avec l'UAC de Windows VISTA.

Schéma

Architecture du système de télédistribution SIG:



Risques identifiés

Application de télédistribution développée en Delphi. Solutions pas forcément accessible dans cet environnement de développement. Traduction nécessaire des API.

Disponibilité de la documentation technique Microsoft.

Annexe B:

COMPOSANTS DU JETON D'ACCES WINDOWS VISTA

Nom	Description
Utilisateur	SID du compte utilisateur ayant ouvert la session.
Groupes	SID et attributs des groupes de sécurité du compte utilisateur.
Logon SID (compris dans la liste des groups)	SID identifiant la demande de login. Plusieurs jetons identiques peuvent être créés pour le même utilisateur. Le logon SID identifie alors les différentes procédures de login. Les processus SYSTEM n'ont pas de Logon SID (sauf Svchost) Un jeton anonyme (avec ID d'authentification = 98) n'a pas de Logon SID. (48)
Privilèges	Liste des privilèges accordés à l'utilisateur.
Groupe principal	SID du groupe principal du propriétaire (requis pour la compatibilité POSIX)
DACL	Le jeton d'accès est un objet sécurisé. Le DACL Contrôle l'accès au jeton d'accès. Le privilège SE_DEBUG_NAME permet de contourner le contrôle d'accès du jeton.
Source	Chaîne de caractères identifiant le processus à l'origine de la creation du jeton.
TokenID	Identifie le jeton de manière unique dans le système
Date et heure d'expiration	Actuellement, Windows ne tient pas compte de ce champ. Ainsi, une session ouverte n'expire jamais, y compris si le compte utilisateur est supprimé !
ID de modification	LUID changeant chaque fois que le jeton est modifié. Permet à un processus de savoir si un jeton a été modifié depuis sa création.
Type	Défini le type de jeton. Primaire ou impersonalisation (emprunt d'identité)
Niveau d'emprunt d'identité	Niveau d'emprunt d'identité du processus lorsque le jeton n'est pas primaire. Les niveaux « d'impersonalisation » possibles sont: • Anonyme • Identification • Impersonalisation • Délégation
ID d'authentification	Cet identificateur, contrairement au Logon SID est identique pour chaque jeton utilisateur représentant un même utilisateur (et un même niveau d'intégrité). Les ID d'authentifications inférieures à 1000 sont réservés. • x03E7 = 999 = SYSTEM • x03E6 = 998 = ANONYMOUS • x03E5 = 997 = LOCAL _SERVICE • x03E4 = 996 = NETWORK_SERVICE L'ID d'authentification est utilisé comme identificateur dans la création de la « Window Station » des services. Par exemple, les services SYSTEMES sont créés dans la « Winsta » Service-0x0-3e7$. (voir chapitre *Winsta*)
ID de session	Identifie la session dans laquelle les processus liés à ce jeton seront créés (voir chapitre *Sessions*).
Jeton lié (nouveau sous Vista)	Pour un jeton administrateur restreint, il s'agit du jeton complet lié (et vice-versa)
Niveau d'intégrité (nouveau sous Vista)	SID indiquant le niveau d'intégrité accordé à l'utilisateur. • System S-1-16-16384 • High S-1-16-12288 • Medium S-1-16-8192 • Low S-1-16-4096

Tableau 10 Composants des principales caractéristiques d'un jeton d'accès sous Windows Vista (49)

Annexe C:

DOSSIERS ET CLES DE REGISTRE MARQUES PAR UN LABEL D'INTEGRITE

Les dossiers et les ruches HKLM et HKCU ont été scannés par l'utilitaire « Acldump.exe » développé lors de ce travail. Le résultat est listé ci-dessous :

C:	Étiquette obligatoire\Niveau obligatoire élevé
C:\$Recycle.Bin	Étiquette obligatoire\Niveau obligatoire faible
C:\ProgramData\Microsoft\Windows\DRM	Étiquette obligatoire\Niveau obligatoire faible
C:\Users\username\AppData\Local\Microsoft\Windows\ History\Low	Étiquette obligatoire\Niveau obligatoire faible
C:\Users\username\AppData\Local\Microsoft\Windows\ Temporary Internet Files\Low	Étiquette obligatoire\Niveau obligatoire faible
C:\Users\username\AppData\Local\Microsoft\Windows\ Temporary Internet Files\Virtualized	Étiquette obligatoire\Niveau obligatoire faible
C:\Users\username\AppData\Local\Temp\Low	Étiquette obligatoire\Niveau obligatoire faible
C:\Users\username\AppData\LocalLow	Étiquette obligatoire\Niveau obligatoire faible
C:\Users\username\Favorites	Étiquette obligatoire\Niveau obligatoire faible
C:\Windows\ServiceProfiles\LocalService\AppData\ LocalLow	Étiquette obligatoire\Niveau obligatoire faible
C:\Windows\ServiceProfiles\NetworkService\AppData\ LocalLow	Étiquette obligatoire\Niveau obligatoire faible
C:\Windows\System32\config\systemprofile\AppData\ LocalLow	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\AppDataLow	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Internet Explorer\ IntelliForms	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Internet Explorer\ InternetRegistry	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Internet Explorer\ LowRegistry	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Internet Explorer\ PageSetup	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Internet Explorer\ Toolbar	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Internet Explorer\Toolbar\ShellBrowser	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Internet Explorer\Toolbar\WebBrowser	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Windows\CurrentVersion\ Explorer\ CLSID\{645FF040-5081-101B-9F08-00AA002F954E}\DefaultIcon	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Windows\CurrentVersion\ Explorer\LowRegistry	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Windows\CurrentVersion\ Explorer\ MenuOrder\Favorites	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Windows\CurrentVersion\ Internet Settings\5.0\LowCache	Étiquette obligatoire\Niveau obligatoire faible
HKCU\Software\Microsoft\Windows\CurrentVersion\ Internet Settings\Passport\LowDAMap	Étiquette obligatoire\Niveau obligatoire faible