Faire un RegFix

Par Roger54, Regis59, Wawaseb, !aur3n7, Sloshy.
(Reproduit avec l'accord de Roger54.)

Ce post a était crée dans le but de faire connaître les bases de la création d'un **RegFix.**
La plupart des informations qu'il contient proviennent de recherches sur Internet.
Si vous pensez que ce tutoriel contient des erreurs ou si des améliorations peuvent être apportées ou des modifications, vous pouvez le signaler.

La suppression ou la modification de clefs de registre n'est pas sans danger.
Mieux vaut prendre des précautions, plusieurs choix s'offrent à nous.

* Faire exporter la clef ou les clefs de registre que l'on modifie.

* Utiliser une commande via un fichier .bat ou directement en ligne de commande.
  Ce qui évite d'intervenir directement dans le registre

* Utiliser un logiciel extérieur comme Erunt

1. Faire exporter la clef ou les clefs de registre que l'on modifie

Pour accéder à la base de registre :
Démarrer, Exécuter, entrer regedit puis OK.
Ou taper regedit.exe dans Rechercher sous Vista, si vous n'avez pas activé le bouton Exécuter.
Choisissez la clef que vous voulez exporter puis cliquez sur l'onglet Registre et Exporter un fichier du Registre…
Donnez un nom à votre fichier et enregistrez-le. Un fichier .reg sera créé à l'emplacement choisi, il suffira de double cliquer dessus puis d'accepter la fusion pour que les clefs soient restaurées.

2. Fichier .bat ou en ligne de commande.

Exemple pour restaurer cette clé :
HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa

Avec un fichier .bat :
Ouvrez le bloc-notes et entrer cette ligne à l'intérieur :
RegEdit /e sauvegarde.reg HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa
Enregistrer votre fichier sous unNom.bat (unNom est modifiable)
Double cliquez dessus, un fichier sauvegarde.reg sera crée à l'emplacement où vous utilisez unNom.bat. Il vous suffira de double cliquer dessus pour restaurer la clef.

En ligne de commande :
Faites démarrer, exécuter puis entrée :
RegEdit /e c:\sauvegarde.reg
HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa
Validez avec ok.
Un fichier sera crée directement à la racine du disque dur c:\sauvegarde.reg.

Avec une invite de commande :
Faite Démarrer, Exécuter puis entrer cmd.
L'invite de commande s'ouvre, vous n'avez plus qu'à insérer la ligne à l'intérieur
RegEdit /e c:\sauvegarde.reg
HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa

3. Utilisation de Erunt.

Logiciel permettant de sauvegarder la base de registre et bien sûr de la restaurer.
Nous avons la possibilité de restaurer la base de registre de différentes façons:

* A partir du répertoire où sont stockées les sauvegardes.
* A partir de la console de récupération Windows

Tutoriel : http://www.vista-xp.fr/forum/topic77.html

Le fichier .reg

Lors des recherches que vous allez effectuer pour savoir si tel ou tel élément est dangereux, soyez sûr de la réponse que vous allez trouver.
Ce n'est pas parce qu'une personne dit que cet élément est mauvais qu'il l'est forcement.
Recherchez plusieurs réponses identiques si possible de helpers ou de sites connus.

Généralement on utilise le bloc-notes de Windows, mais tout éditeur de texte fera l'affaire.
On ouvre notre éditeur de texte, on tape notre script puis on l'enregistre sous un nom avec l'extension .reg, comme ceci:
windows.reg (windows étant juste un exemple)

Note **: Toujours décocher le retour automatique à ligne dans l'éditeur de texte**.
Explication : si la ligne est "trop longue" et que le retour automatique est activé, le texte final sera interprété comme 2 lignes distinctes.
Au mieux il ne se passera rien, au pire ce qui se passera n'aura aucun rapport avec ce que vous vouliez faire et peut être catastrophique.
Si le retour est désactivé, il n'y aura aucune marque de retour à la ligne décelée dans l'interprétation et les 2 lignes seront vues en continue (plus exactement là où vous voyez 2 lignes, l'interpréteur n'en verra qu'une car il ne voit la fin d'une ligne que quand il rencontre la marque de retour à la ligne, conformément au souhait. L'ordre sera bien interprété et exécuté.

Voici l'icône que vous devriez voir apparaître :

http://i28.servimg.com/u/f28/11/05/93/83/iconer10.jpg

Qu'est-ce qu'une clef de registre ?

Voici un exemple:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
"avgnt"="C:\Program Files\Avira\AntiVir PersonalEdition Classic\avgnt.exe"
"1465c3cf"="C:\WINDOWS\system32\krlfwmfx.dll"
"Microsoft Windows Updeta"="dyvhkp.exe"
```

Vous avez:

* La clé racine en rouge (ou ruche)
* Les sous-clés en vert
* Le nom de la valeur en bleu
* La donnée de la valeur en violet

Remarque :
Les ruches sont au nombre de 5 (visibles)
Les clés et sous clés sont visibles à gauche dans l'interface de la base de registre ou registry.
On parle de déplier.
Dans notre exemple on ouvre la ruche HKEY_LOCAL_MACHINE
Puis la clé Software, puis la sous-clé Microsoft etc. pour arriver à lire ce qui se situe comme valeurs dans la sous-clé Run.
Le nom de la valeur se situe à gauche de la partie droite de l'interface.
La valeur se situe ensuite il suffit de cliquer dessus pour la modifier manuellement.

Dans nos scripts on ajoute des crochets autour de la clé de registre (ruche + sous-clé), comme ceci.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"avgnt"="C:\Program Files\Avira\AntiVir PersonalEdition Classic\avgnt.exe"
"1465c3cf"="="C:\WINDOWS\system32\krlfwmfx.dll"
"Microsoft Windows Updeta""dyvhkp.exe"
```

Lors des prochains Regfix que vous allez effectuer, vous allez agir soit sur les sous-clés soit sur la valeur, son nom ou sa donnée.

Maintenant on va donc apprendre à supprimer une sous-clé ou à supprimer une valeur.

Notions préliminaires:

Mais comment sait-on ce qu'il faut supprimer dans une clef?
...
Eh bien il faut chercher, exemple:

```
HKEY_CLASSES_ROOT\CLSID\{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}\InProcServer32
@="C:\WINDOWS\system32\qhcvdw.dll"
```

On commence par rechercher le dernier élément de la clef, c'est à dire la donnée, puis l'élément précédant et ainsi de suite.

1. Faut-il supprimer la donnée de @ : **C:\WINDOWS\system32\qhcvdw.dll**
Recherche avec Google : néfaste, donc à supprimer

2. InprocServer32 : rien de probant, on le retrouve dans beaucoup de clés.

3. Cette partie : {c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}
C'est un CLSID, vous en avez s'en doute déjà vu. (**CL**as**S ID**entifier ou **GUI**, **G**lobal **U**nique **I**dentifier)
Recherche avec Google : ce CLSID est mauvais, à supprimer.

4. HKEY_CLASSES_ROOT\CLSID\, cette partie est légitime. Vous pouvez la retrouver sur votre PC.

En résumé :
La valeur et mauvaise ainsi que le CLSID.
On va devoir supprimer les deux alors !
Eh bien ça ne serait pas faux, mais en fait ce n'est pas nécessaire. Nous allons voir pourquoi.

Il faut voir cette clé (et toutes les clés) comme un arbre:
Le tronc: HKEY_CLASSES_ROOT
Une des grosses branches qui part du tronc: CLSID
...
Et ainsi de suite jusqu'à arriver à la plus petite branche :
"C:\WINDOWS\system32\qhcvdw.dll"

Donc si on coupe le tronc il n'y a plus rien, si on coupe une grosse branche toutes les plus petites branches tomberont avec...
Dans notre cas, on a donc juste besoin de supprimer le CLISD :
{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888} ce qui supprimera également la valeur.
InProcServer32 étant lié à {c7cd9e83-3bf6-47f8-b2e2-b114c96c1888} et @="C:\WINDOWS\system32\qhcvdw.dll" étant une valeur de InProcServer32.

Toutes les clefs commençant par
HKEY_CLASSES_ROOT\CLSID\{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}
Seront supprimées.
Il peut y avoir plusieurs clefs :

HKEY_CLASSES_ROOT\CLSID**{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}\InProcServer32**
@="C:\WINDOWS\system32\qhcvdw.dll"

HKEY_CLASSES_ROOT\CLSID**{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}\ProgID**
@="DAO.DBEngine.35"

Tout ce qui est en gras sera supprimé.

Comment fait-on pour supprimer ce CLSID:

Voici la syntaxe pour la clef:

[-HKEY_CLASSES_ROOT\CLSID\{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}]

Qu'est-ce que l'on a fait:
On a ajouté le signe moins "-" entre le premier crochet et le début de la clef.
Ce signe permet de supprimer ce qu'il y a au bout de la chaîne, dans notre exemple on devrait donc retrouver la valeur "c7cd9e83-3bf6-47f8-b2e2-b114c96c1888", ce qui est bien le cas.
Si on avait fait ça:
[-HKEY_CLASSES_ROOT\CLSID]
On aurait supprimé tous les CLSID et tout ce qui va avec, à ne surtout pas faire sous peine de gros problèmes par la suite.
Donc être bien sûr de la partie que l'on supprime si on ne veut pas faire une catastrophe.

Et si on avait eu besoin de supprimer que la valeur:

Considérons {c7cd9e83-3bf6-47f8-b2e2-b114c96c1888} comme légitime (ce qui n'est pas le cas ici)
Voici la syntaxe pour supprimer une valeur.

[HKEY_CLASSES_ROOT\CLSID\{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}\InprocServer32]
@=-

Cette fois si, on a remplacé la donnée de la valeur par un signe -.
Cette syntaxe a pour conséquence de supprimer toute la valeur (son nom, son type, sa donnée).

Mais pourquoi supprimer cette valeur alors que l'on aurait eu le même résultat en supprimant simplement InprocServer32 comme ceci :

[-HKEY_CLASSES_ROOT\CLSID\{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}\InprocServer32]

Il peut y avoir des cas ou la clef est légitime mais où la valeur est modifiée par l'infection, il faudra donc remettre la valeur par défaut (nous verrons ça dans la seconde partie).
D'autres cas plus fréquents où il existe plusieurs valeurs sous une même sous-clé et dont au moins l'une d'entre elles est à supprimer.
Ça tombe bien on a un exemple tout en haut, on le remet ici :

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
"avgnt"="C:\Program Files\Avira\AntiVir PersonalEdition Classic\avgnt.exe"
"1465c3cf"="C:\WINDOWS\system32\krlfwmfx.dll"
"Microsoft Windows Updeta"="dyvhkp.exe"

On effectue une recherche google et on s'aperçoit qu'il a deux valeurs à supprimer, le reste ne pose pas de problème:
"1465c3cf"="C:\WINDOWS\system32\krlfwmfx.dll"
"Microsoft Windows Updeta"="dyvhkp.exe"

Dans ce cas voici la syntaxe:

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"1465c3cf"=-
"Microsoft Windows Updeta"=-

On peut donc de cette façon supprimer plusieurs valeurs sans pour autant les supprimer toutes.

Quelques cas spéciaux:

Les explications pour ces 2 cas seront traitées dans la seconde partie.

[HKEY_LOCAL_MACHINE\software\microsoft\windows nt\currentversion\windows]
"appinit_dlls"=c:\windows\system32\vtsttrp.dll

Ce qu'il faut appliquer :

[HKEY_LOCAL_MACHINE\software\microsoft\windows nt\currentversion\windows]
"appinit_dlls"=""

Et

[HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa]
"Authentication Packages"= msv1_0 C:\WINDOWS\system32\awtst.dll

Ce qu'il faut appliquer :

[HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa]
"Authentication Packages"=hex(7):6d,73,76,31,5f,30,00,00

Script final

Vous connaissez maintenant la syntaxe pour supprimer une clef ou une valeur mais vous n'avez toujours pas vu de script entier.
Voila sous quelle forme il se présente :

```
REGEDIT4

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"1465c3cf"=-
"Microsoft Windows Updeta"=-

[-HKEY_CLASSES_ROOT\CLSID\{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}]
```

Il faut donc ajouter **REGEDIT4**, laisser une ligne vide entre REGEDIT4 et la clé, entre les clés elles-mêmes et à la fin.
(Toujours REGEDIT4 en majuscule sinon ça ne fonctionnera pas)
Respecter toujours cette syntaxe, ne pas oublier les crochets autour des clés, bien mettre les "" autour du nom de la valeur.
Indiquer toujours les ruches avec leur nom entier ex : HKEY_LOCAL_MACHINE au lieu de HKLM.
Indiquer également le chemin de la clé en entier, ne pas utiliser de nom court avec des ~.

Note:
Il n’est pas obligatoire de laisser une ligne vide entre chaque clé même si Microsoft indique que c'est nécessaire.

Explication plus en détail du script:

```
REGEDIT4

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"1465c3cf"=-
"Microsoft Windows Updeta"=-

[-HKEY_CLASSES_ROOT\CLSID\{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}]
```

REGEDIT4 correspond à la "Version Éditeur Registre", l'éditeur de registre étant regedit.exe.
La ligne vide: Elle identifie le début d'un nouveau chemin d'accès de Registre.
Les crochets [...] autour de la clef sont des délimiteurs.
Les guillemets "" permettent d'indiquer au Registre que nous avons à faire à une chaîne de caractères et ainsi par exemple lui permettre de gérer les différents espaces qu'il va rencontrer.
Nous verrons par la suite que toutes les données de valeur de type "REG_SZ" doivent avoir des guillemets autour d'elles.

Nous avons vu également dans la première partie une clé de cette forme:

```
HKEY_CLASSES_ROOT\CLSID\{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}\InProcServer32
 @="C:\WINDOWS\system32\qhcvdw.dll"
```

Il y a un @ qui remplace le nom de la valeur.
En fait, nous ne sommes pas obligés de donner de nom à une valeur.
Dans ce cas, lorsque l'on exporte cette clé par exemple, un nom pour cette valeur est crée par défaut.
Ce nom est @.
Il n’est pas nécessaire de mettre des guillemets autour.

Différence entre le type Reg4 et le type Reg5

Jusqu'ici nous avons vu que pour faire un script on utilisait REGEDIT4 comme Version Éditeur Registre, nous allons voir une autre version.

REGEDIT4 fonctionne sous Windows 98, Windows NT 4.0, Windows 2000, Windows Server 2003, Windows XP et Windows Vista.

Il existe également "Windows Registry Editor Version 5.00" (bien respecter les majuscules et minuscules), correspond au type Reg5.
Lui fonctionne sous Windows 2000, Windows Server 2003, Windows XP et Windows Vista.
Si on reprend notre exemple, voici ce qu'il faudrait écrire:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"1465c3cf"=-
"Microsoft Windows Updeta"=-

[-HKEY_CLASSES_ROOT\CLSID\{c7cd9e83-3bf6-47f8-b2e2-b114c96c1888}]
```

La différence entre les deux est au niveau du codage. Le type Reg5 utilise Unicode alors que le type Reg4 est un texte ANSI (code ASCII étendu ANSI).
Nous verrons par la suite que cette différence à des conséquences.

Modification d'une valeur

Dans un premier temps nous allons nous attardez sur le "type" que peut prendre une valeur.

Il existe un grand nombre de "type" différent, voici les plus utilisés:

REG_SZ
REG_BINARY: hex
REG_EXPAND_SZ: hex(2)
REG_MULTI_SZ: hex(7)
REG_DWORD: DWORD

Petite liste de différents types possibles en bas de la page
Nous allons maintenant apprendre à modifier la donnée d'une valeur pour les différents types ci-dessus.
Cette différence va engendrer une petite modification dans notre script, il va falloir maintenant indiquer le type.

1. REG_SZ:

Ce type correspond aux chaînes de texte de longueur fixe.

Avec le type REG_DWORD, c'est le plus utilisé. Ici on a de la chance, pas besoin d'indiquer le type dans le script.
Si on ne donne aucunes indications, c'est le type par défaut qui sera utilisé, celui-ci étant REG_SZ.

Exemple que l'on peut rencontrer dans un script:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion]
"CommonFilesDir"="C:\\Program Files\\Fichiers communs"
```

Ou

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion]
"CommonFilesDir"="\"C:\\Program Files\\Fichiers communs\""
```

Ce que l'on voit dans le registre pour la donnée:
C:\Program Files\Fichiers communs et "C:\Program Files\Fichiers communs"

La donnée est une chaîne de caractères, il faut donc ajouter des guillemets autour d'elle.
On s'aperçoit que les \ sont doublés dans la donnée.
Regedit ne voit pas le \ comme un séparateur de répertoire traditionnel. Au contraire, il traite les \ comme caractère spécial possédant une autre fonction. C'est ce que l'on appelle un caractère d'échappement, il va nous permettre d'afficher littéralement des " ou des \, ceux-ci ayant déjà une autre fonction pour Regedit.
Donc dans une chaîne de caractères si vous voulez afficher un \ ou un ", il faut les faire précéder du caractère d'échappement \ (anti-slah ou backslash).
Si vous oubliez de mettre soit un " soit un \ après le 1er \, le script ne fonctionnera pas.
Note: Pour les chemins UNC qui commencent par 2 backslashes \\, il faudra en indiquer 4 dans le script, comme ceci: "Logonserver"="\\\\servera».

Voici un autre exemple, que l'on a vu dans la première partie :

[HKEY_LOCAL_MACHINE\software\microsoft\windows nt\currentversion\windows]
"appinit_dlls"=c:\windows\system32\vtsttrp.dll

Ici le nom de la valeur est légitime, mais la donnée n'est pas bonne.
Il faut donc remettre sa valeur par défaut.
On fait une petite recherche sur Google ou sur notre propre registre, et on s'aperçoit que la donnée de appinit_dlls est nulle.
Il suffit de ne rien indiquer entre les guillemets.

REGEDIT4

[HKEY_LOCAL_MACHINE\software\microsoft\windows nt\currentversion\windows]
"appinit_dlls"=""

Pas de différence avec le type Reg5.

2. REG_BINARY:

Ce type permet de gérer dans le registre des données binaires. Les données binaires se composent d'une suite d'octets qui peut être cryptée. La plus part des informations concernant les composants matériels sont stockés dans ce format.

Dans le script :

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\GBG]
"GrafBlumGroup"=hex:7d,0f,ac,aa,95,c0,c0,4a,bd

Dans le registre :

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\GBG]
"GrafBlumGroup"=7d,0f,ac,aa,95,c0,c0,4a,bd

Ici ce n'est plus une valeur de type REG_SZ, dans ce cas Regedit est un peu moins intuitif.
A droite du signe égal il va falloir désigner le type de la valeur : ici **hex : et la donnée**

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\GBG]
"GrafBlumGroup"=*hex:7d,0f,ac,aa,95,c0,c0,4a,bd*

Ici ce sont des valeurs hexadécimales, qui sont interprétées comme telles. Elles sont séparées par une virgule.

Note : Hexadécimal pour 6 lettres (abcdef) et 10 chiffres (0123456789)
Ne pas oublier les **:** après hex et pas d'espace avant la valeur

REGEDIT4

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\GBG]
"GrafBlumGroup"=hex:7d,0f,ac,aa,95,c0,c0,4a,bd

Pas de différence avec le type Reg5.

3. REG_EXPAND_SZ:

Ce type permet de gérer une chaîne de caractères comportant des variables d'environnements système comme, %SystemDrive%, %SystemRoot%, %windir%...

La déclaration du type ce fait comme ceci hex(2):

Dans le script:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion]
"DevicePath"=hex(2):25,53,79,73,74,65,6d,52,6f,6f,74,25,5c,69,6e,66,3b,25,53,\
79,73,74,65,6d,44,72,69,76,65,25,5c,49,6e,73,74,61,6c,6c,5c,70,69,\
6c,6f,74,65,73,00
```

Dans le registre:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion]
"DevicePath"="%SystemRoot%\inf;%SystemDrive%\Install\pilotes"
```

Ouais et comment on fait là ?

La donnée, la voici dans le registre: %SystemRoot%\inf;%SystemDrive%\Install\pilotes
Et dans le .reg:
25,53,79,73,74,65,6d,52,6f,6f,74,25,5c,69,6e,66,3b,25,53,\
79,73,74,65,6d,44,72,69,76,65,25,5c,49,6e,73,74,61,6c,6c,5c,70,69,\
6c,6f,74,65,73,00

Comment fait-on pour passer de l'un à l'autre?

Comme vous le savez un ordinateur c'est pas très fute fute. Bon il se débrouille quand même en binaire et en hexadécimal.
Il ne connait pas les lettres, pour afficher une lettre il a besoin d'un nombre.
Tel nombre affiche tel caractère.
Il existe des tables de correspondance créées spécialement pour ça.
La table ASCII, l'ASCII prolongé en ANSI (avec quelques caractères en plus), l'UNICODE ...
http://www.table-ascii.com/
Comme nous l'avons vu précédemment, le type Reg4 fonctionne avec des textes ANSI tandis que le type Reg5 l'UNICODE.
La différence entre les 2 c'est qu'avec l'ASCII donc L'ANSI, un caractère est codé sur 1 octet tandis qu'avec l'UNICODE ce même caractère sera codé sur 2 octets.
Ainsi la lettre "A" (en majuscule, il y a une autre valeur pour le "a" minuscule) est donc 0x41 dans le premier cas et
0x0041 dans le second. (Le **0x** indique que c'est une valeur donnée en hexadécimal)

Note: Lorsque vous entrez une lettre ou un signe dans un fichier texte, Excel ou même sur ce forum, vous appuyer directement sur la touche de votre clavier, mais il est possible de l'entrer en tapant directement le code qui lui correspond.
Faite alt + n°code(en décimal) et la lettre ou le symbole s'affiche.

Une fois que l'on sait ça, on peut facilement finir notre script.
25 -> %
53 -> S
79 -> y

```
REGEDIT4

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion]
"DevicePath"=hex(2):25,53,79,73,74,65,6d,52,6f,6f,74,25,5c,69,6e,66,3b,25,53,\
79,73,74,65,6d,44,72,69,76,65,25,5c,49,6e,73,74,61,6c,6c,5c,70,69,\
6c,6f,74,65,73,00


Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion]
"DevicePath"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,00,74,00,\
25,00,5c,00,69,00,6e,00,00,66,00,3b,00,25,00,53,00,79,00,73,00,74,00,65,00,6d,00,44,00,72,\
00,69,00,76,00,65,00,25,00,5c,00,49,00,6e,00,73,00,74,00,61,00,6c,00,6c,00,5c,00,70,00,69,\
00,6c,00,6f,00,74,00,65,00,73,00,00,00
```

Cette fois si notre \ permet de passer à la ligne sans pour autant couper notre donnée.

Si vous regardez bien et que vous comptez bien, vous allez apercevoir un caractère en plus à la fin de la chaîne, " 00 ", c'est un caractère de terminaison (caractère NULL) qui permet de dire que la chaîne est terminée.
Cela implique que nous ne pouvons pas insérer de 00 à l'intérieur de notre chaîne sinon celle-ci sera considérée comme terminée.
Il existe également un caractère de terminaison NULL pour le type REG_SZ mais celui-ci est ajouté automatiquement.

4. REG_MULTI_SZ:

Ce type est utilisé pour représenter des valeurs qui contiennent une liste de données ou plusieurs données.

REG_MULTI_SZ fonctionne de la même façon que REG_EXPAND_SZ, la seule différence vient dans la déclaration du type, au lieu de mettre hex(2) il faut inscrire hex(7).
Le caractère de terminaison est différent aussi, à la place de " 00 ", on a " 00,00 " . On double le caractère NULL.
Note: Contrairement au cas précédant nous pouvons ici intégrer des caractères NULL, 00, dans notre chaîne puisqu'il en faut 2 de suite pour indiquer la fin de la chaîne.

Nous allons voir le cas traité dans la première partie et que l'on a vu sur les forums.
(Infection Vundo)

```
[HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa]
"Authentication Packages"= msv1_0 C:\WINDOWS\system32\awtst.dll
```

Ici la valeur de la clef à était modifiée ; voici la valeur d'origine :
:
HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa]
"Authentication Packages"= msv1_0

Il faut donc supprimer tout ce qu'il y a après msv1_0.
Comme c'est une valeur de type REG_MULTI_SZ nous fonctionnons en hexadécimal avec hex(7) à mettre dans le script.

Vous savez comment ça fonctionne maintenant

6d -> m
73 -> s...

```
REGEDIT4

HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa]
"Authentication Packages"=hex(7):6d,73,76,31,5f,30,00,00
```

On fini donc la valeur par le caractère de terminaison 00,00.

```
Windows Registry Editor Version 5.00

HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa]
"Authentication Packages"=hex(7):6d,00,73,00,76,00,31,00,5f,00,30,00,00,00,00,00
```

5. REG_DWORD:

Les entrées de ce type peuvent contenir des informations sur les pilotes des périphériques, des valeurs booléennes (0 ou 1), des quantités (par exemple, le nombre de secondes qui peuvent s'écouler avant que quelque chose se passe ou pas)...

La déclaration du type ce fait comme ceci dword:
Il n'y a pas de caractère de terminaison pour ce type.
La valeur est exprimée en hexadécimal.

Que veut dire dword ?

Un nombre binaire de 4 bits s'appelle un quartet.
Un quartet permet d'écrire 16 nombres de 0 à 15
Hexadécimal: 0x0 à 0xF

Un nombre binaire de 8 bits s'appelle un octet (byte en anglais)
Un octet permet d'écrire 256 nombres de 0 à 255
Hexadécimal: 0x00 à 0xFF

Un nombre binaire de 16 bits est généralement appelé mot "word " en anglais.
Un tel groupement de 16 bits permet d'écrire 65 536 nombres de 0 à 65 535.
Hexadécimal: 0x0000 à 0xFFFF

Un nombre binaire de 32 bits est généralement appelé double mot " double word " ou " dword " en anglais.
Il permet d'écrire les nombres de 0 à 4 294 967 295
Hexadécimal: 0x00000000 0xFFFFFFFF

Un nombre binaire de 64 bits est généralement appelé quatre mot " quad word " ou " qword " en anglais.
Il permet d'écrire les nombres de 0 à 18 446 744 073 709 551 615
Hexadécimal: 0x0000000000000000 0xFFFFFFFFFFFFFFFF

La valeur dans la clef aura donc une expression similaire à celle-ci: 0x00000000.
On l'a note dword:00000000

Dans le script :

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\SspiCache\digest.dll]
"Capabilities"=dword:00004050
"RpcId"=dword:0000ffff

Dans le registre :

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\SspiCache\digest.dll]
"Capabilities"=0x00004050 (16464)
"RpcId"=0x0000ffff (65535)

```
REGEDIT4

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\SspiCache\digest.dll]
"Capabilities"=dword:00004050
"RpcId"=dword:0000ffff
```

Pas de différence avec le type Reg5.

Note:
Nous ne sommes pas obligés d'indiquer les zéros de têtes, Regedit le fera tout seul pour nous.
Si on affiche ceci dans le scrïpt
"Capabilities"=dword:4050
Voici ce que l'on verra dans le registre
"Capabilities"=0x00004050 (16464)

Voici quelques autres types (voir les mêmes) et la façon dont ils sont représentés dans le registre:
REG_NONE : hex(0)
REG_SZ : hex(1)
REG_BINARY : hex(3) idem que hex
REG_DWORD ou REG_DWORD_LITTLE_ENDIAN (pour XP): hex(4)
REG_DWORD_BIG_ENDIAN (pour XP): hex(5)
REG_LINK : hex(6)
REG_RESOURCE_LIST : hex(8)
REG_FULL_RESOURCE_DESCRIPTOR : hex(9)
REG_RESOURCE_REQUIREMENTS_LIST : hex(a)
REG_QWORD : hex(b)