

MicroWinpeBuilder

Apprendre à adapter soi-même un Winpe10 64bits en ajoutant, par exemple, MMC (devices, disks, services, firewall), le bureau avec la barre des tâches, ncpa, netsh, wifi, MSI, le serveur de fichiers, audio, wow64, session adm, bits, winrm, imprimante/scan usb ou réseau ...

Un ensemble de scripts écrits en Powershell illustre cet apprentissage.

Ce document sera éventuellement utile aux débutants curieux comme moi. Je doute qu'un expert y trouve un quelconque intérêt.

C'est maintenant à vous de compléter et corriger ce document.

Présentation du but poursuivi

- Comprendre comment adapter Winpe10 généré par l'ADK
- Apprendre à réaliser les investigations de base
- Intégrer les informations découvertes ou mises à disposition sur internet
- Apporter un complément pour le débutant.

Mise en garde

Les scripts proposés ici permettent de construire un WinPe. Mais en raison de la technologie des scripts PS basés sur DotNet, la taille du fichier Boot.wim est voisine de 600Mo. De plus le démarrage de PowerShell est lent.

Il s'agit d'une construction à but pédagogique.

Les contraintes

Puisqu'il s'agit d'apprendre et de comprendre, il ne faut pas masquer les moyens utilisés.

- Règle 1 : ne pas employer de logiciels externes (donc sans «Winbuilder »)
- Règle 2 : utiliser uniquement des scripts PowerShell (incluant du c# si besoin)

Le principe

J'ai collecté un ensemble d'informations dans diverses sources mises à disposition sur internet (comme Kullenen_Ask : <http://www.msfn.org/board/topic/143241-portable-windows-7-build-from-winpe-30> et quelques autres) et principalement sur le site TheOven (<http://theoven.org>).

L'apprentissage consiste à lire et assimiler ces informations. Puis à les mettre en oeuvre pour construire « son » WinPe.

Pour illustrer cet apprentissage et montrer la pertinence de ces informations, j'ai écrit quelques scripts PS. Ils permettent d'obtenir un WinPe rudimentaire mais qui fonctionne. Ils utilisent en entrée le fichier boot.wim créé par l'ADK. Vous pourrez donc ajouter vos pilotes avec DISM à tout moment.

Ensuite à vous de chercher comment intégrer d'autres composants, à vous de créer vos propres scripts, de modifier les scripts PS fournis avec MicroWinpeBuilder.

L'investigation et la mise en oeuvre des résultats

L'investigation conduit à identifier des objets (fichiers, clés de registre, ACL) qu'il faudra injecter dans le fichier « Boot.Wim » de Winpe.

Elle permet :

- de construire des listes d'éléments à ajouter
- d'adapter les scripts selon son envie et ses découvertes.

Jusqu'à présent, je déposais manuellement les résultats de mes investigations. Maintenant, les scripts PS injectent les modifications pendant la mise au point. Et les mêmes scripts permettent aussi de construire le WinPe final.

Et le plus important est que le fichier de départ est toujours le même, le fichier boot.wim créé par l'ADK. C'est ce qui assure la reproductibilité des tests.

Une comparaison rapide avec la référence WinPeSE

WinPeSe utilise le fichier ISO de Win10Entreprise Evaluation. Il n'utilise pas ADK (hormis quelques outils qui sont directement téléchargés si j'ai bien compris).

Ce fichier ISO contient deux fichiers essentiels :

- Boot.wim, le WinPe de MS utilisé lors de l'installation de windows10.
- Install.wim, qui est l'image de windows10

WinPeSe capture donc ces deux fichiers. Il en extrait les fichiers nécessaires dont les listes sont contenues dans les scripts de Winbuilder. Les ruches sont copiées et modifiées. Tous ces éléments sont ajoutés au répertoire « target ».

WinPeSe apporte aussi ses propres fichiers et programmes dans ses scripts (code binaire encodé sur 64 bits?). Puis Boot.wim est assemblé.

C'est de la précision et du grand art. Mais avant cela il a fallu trouver toutes les modifications à traiter. L'automatisme du programme masque la complexité du travail préalable et la connaissance nécessaire pour construire ce WinPe. Les scripts de WinPeSe sont bien sûr riches en détail et apportent un éclairage sur les points les plus obscurs et complexes.

Ma seule remarque : il m'a été impossible de modifier ultérieurement le fichier Boot.wim de WinpeSe avec DISM pour ajouter un « package » de l'ADK comme dotnet. Dommage.

Sites :

<http://wb.paraglidernc.com/Help/default.html> et <http://www.paraglidernc.com/winbuilder/default.htm>

Ce qui est supposé connu

- avoir lu la documentation de WinPe disponible sur les sites de MS
- être familier avec les outils disponibles dans l'ADK et savoir utiliser DISM, BCDEDIT, BCDBOOT, etc.
- savoir modifier un BCD

ne pas oublier que le BCD pour WinPe contient les clés pour le RamDisk qui ne sont pas présentes dans le BCD du disque dur après une installation normale de windows 10

normal

- savoir construire les BCD pour des VHD.
Les VHD sont très pratiques pour les investigations car les fichiers déposés persistent après le reboot. Les modifications des ruches ne persistent pas.
- avoir lu les scripts CopyPe.cmd et MakeWinpeMedia.cmd de l'ADK

Structure du document

Une première partie rassemble toutes les informations utiles relatives à WinPe.

La seconde partie donne quelques idées pour se lancer dans l'investigation.

La troisième partie fournit une illustration mise en œuvre par les scripts.

Trop de détails tuent l'information

Ce document ne reprend pas tous les détails (clés, fichiers, acl) présents dans les scripts PS. Reportez vous et lisez les scripts si vous voulez vérifier la présence d'un fichier, d'une clé...

Partie Connaissances de WinPe

La modification des ruches : « C : » remplacé par « X : »

Différents sites indiquent que cette modification (format texte et binaire !) est obligatoire. Mais depuis au moins le build 10586, mes scripts ne procèdent pas à ce remplacement et cela fonctionne.

Le minimum à savoir concernant le bureau Explorer.exe

Pas très beau mais suffisant

Pour obtenir un bureau avec un certain confort lors de la manipulation des fenêtres, il faut mettre en œuvre le gestionnaire de fenêtres « Dwm.exe ». Sans « Dwm » et avec le ruban des fenêtres « explorer », il apparaît une bande noire perturbatrice au dessus du ruban.

« DWM » nécessite le service CoreMessagingRegistrar pour le build 10240 et aussi le pilote WindowsTrustedRt pour le build 10586 (voir Win10PeSe).

Comment ont ils trouvé ? Je ne sais pas !

Avec un Winpe actif, la ruche « Default » dans regedit correspond au fichier « ...\\config\\default ». Elle est utilisée par le compte « System » pour devenir « HKCU ». Diverses clés utilisées par « explorer » doivent être injectées pour obtenir un fonctionnement sans anomalies.

...\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\UserSignedIn=1 pour éviter par exemple le délai avant l'apparition des icônes de la barre de tâche.

Sans le service « Themes », je n'ai pas réussi à afficher une image de fond d'écran après le lancement de « explorer.exe ». Ce service nécessite DWM.

Barre de « titre » en couleur pour la fenêtre active :

la clé suivante semble suffisante : HKLM\\...\\DWM\\ColorPrevalence = 1

L'affichage du « Wallpaper » reste encore un peu obscur pour moi.

Le minimum à savoir sur Explorer.exe

« Explorer.exe » réalise deux fonctionnalités différentes :

- il crée un bureau pour l'utilisateur qui ouvre une session
- il permet d'explorer le système de fichiers du PC

La seconde fonctionnalité est disponible dans un Winpe original. Le fichier « explorer.exe » est absent mais la fonctionnalité est active. Pour le vérifier, dans un Winpe original actif, ouvrir « Notepad », menu « fichier/ouvrir » et on obtient un explorateur de fichiers qui permet presque toutes les actions sur le système de fichiers. Seule sa mise en œuvre est inconfortable sans un bureau.

La première fonctionnalité, le bureau, nécessite le fichier « explorer.exe ». Si on lance ce programme depuis une boîte « Cmd » et en traçant avec « procmon », on peut retrouver les éléments manquants.

Ensuite , il faut que Winpe lance automatiquement « explorer » au démarrage. Pour cela, il faut ajouter la clé « ...\\windows Nt\\winlogon\\shell = explorer.exe ». Lorsque « explorer.exe » est lancé pour la

première fois, il teste la présence de cette clé et installe le bureau de l'utilisateur. Lors du second lancement, il ouvre une fenêtre pour explorer le système de fichiers.

« Explorer » met en œuvre un ensemble d'éléments COM. Le dialogue COM utilise des clients et des serveurs COM. Le dialogue COM met en oeuvre des autorisations selon la configuration des éléments COM définie dans le registre. Pour « explorer », de nombreux composants COM sont configurés pour demander une autorisation. Mais WinPe (WinRe?) ne met pas en œuvre de mécanisme permettant de répondre à ces demandes d'autorisation.

Pour contourner cette sécurité, il faut modifier les valeurs «HKCR\APPID\Runas » pour les objets COM (tous de préférence).

Ce point reste mystérieux. Mais si je comprends bien, l'absence du paramètre « runas » indique au serveur COM qu'il n'a pas besoin de vérifier l'identité de l'utilisateur connecté. Si ce paramètre est présent (et égal à InterActive), le serveur contrôle l'identité de l'utilisateur. Or, ce contrôle échoue avec WinPe car le compte « system » n'est pas un compte ordinaire.

Pour apprendre plus sur la sécurité dans COM :

[https://msdn.microsoft.com/en-us/library/ms682359\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms682359(v=vs.85).aspx)

[https://msdn.microsoft.com/en-us/library/ms680046\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms680046(v=vs.85).aspx)

La fonction « Affichage du bureau »

La source découverte tardivement : <http://theoven.org/index.php?topic=1722.msg20025#msg20025>

J'aurais dû commencer par là. Mais voilà, comment connaître ? D'où l'intérêt d'un effort de centralisation de la documentation bien que les évolutions soient très rapides.

L'activation de cette fonction :

- avec les touches « Windows » + D
- en cliquant dans le coin en bas à droite (1 mm de large) entre l'horloge et le bord de l'écran
- en utilisant le menu contextuel de la barre de tâche

Dans Winpe :

La fenêtre « Shell_TrayWnd » reçoit les messages (WM_XXXX) envoyés par les diverses sources d'activation mais ne les traite pas.

Un peu d'investigation avec SPY++

Spy++ (indispensable et présent dans visual studio community gratuit) montre les messages suivants reçus par « Shell_TrayWnd » :

Msg 0x579 wParam 3 lParam 0

Msg 0x43F

Dans un windows10 :l'envoi du msg 0x579 (wParam 3 lParam 0) suffit pour basculer l'affichage du bureau.

Idée : tenter de comprendre pourquoi « explorer » ne traite pas ces messages en utilisant windbg.

Correction dans Winpese :

Le programme Wind.exe et la dll MsgHook.dll assure le traitement attendu.

Algo simplifié de ces fichiers : en gros, lors de la réception du bon message dans le hook, réduire ou agrandir les fenêtres principales pour l'affichage du bureau ou le retour à l'affichage précédent.

Note : comment est lancé ce programme Wind.exe/Myhook.dll dans WinpeSe?

Le fichier « users\default\ntuser.dat » est utilisé par la session « system ». Il est aussi recopié pour être utilisé par la session adm.

La clé runonce de ce fichier contient plusieurs valeurs permettant le lancement de programmes.

Et la valeur « autorun = autorun.exe » permet de traiter le fichier autorun.ini contenant les commandes à lancer, dont « Wind.exe »

Modification de la taille de l'écran

Lors de la modification d'un paramètre comme la taille de l'écran, le programme responsable de ce changement envoie un message « WM_DISPLAYCHANGE » à toutes les fenêtres principales. Et chacune agit en conséquence et se redimensionne. Mais dans Winpe, il semble que Explorer ne traite pas ce message. Mais il traite le message « WM_SETTINGCHANGE » qui conduit aussi à redimensionner la fenêtre du bureau, et donc toutes les autres fenêtres.

Solution pour mon script de changement de la taille de l'écran :

envoyer le message « WM_SETTINGCHANGE » à la fin du redimensionnement de l'écran

Solution de WinpeSe avec les programmes « FixScreen (JFX) » et Broadcast EnvChange (Galapo)» :

boucle : attendre le message « WM_DISPLAYCHANGE » et envoyer le message « WM_SETTINGCHANGE »

Mon script pour modifier la taille de l'écran : IHM_Get-Set_WinpeScreenResolution.ps1

Pour lire la configuration vidéo en PS :

```
Add-Type -AssemblyName system.windows.forms  
[system.windows.forms.Screen]::allscreens
```

Anomalies

Anomalies corrigées dans mon winpe 1511(build10586), pour mémoire !

- Wpeinit se bloque 5 minutes au démarrage de winpe

La clé « policymanager.dll » était présente mais la clé software\micros...\policymanager » était absente.

- Le démarrage du service « coremessagingregistrar » échouait
il manquait la clé « software\micros...\securitymanager »
- Les icônes du bureau s'affichaient une minute après le bureau
il manquait des clés dans « HKCU\..\explorer »
- Le son ne fonctionnait pas et l'icône de notification en bas à droite affichait « pas de hauts parleurs »
une ACL interdisait l'accès à la clé « ...\MmDevices\Audio\Render...\properties » pour les comptes utilisés par le service AudioSrv.
- Impossible de déplacer les icônes du bureau
cela provenait de la clé software\microsoft\ole qui ne contenait pas les informations liées au DragAndDrop. Le chargement des nouvelles valeurs échouait en raison des Acl de la clé.
- Création de raccourci impossible sur le bureau
il manquait les fichiers « appwiz.cpl » et « osbaseln.dll » ainsi que leurs « .mui »
- Pas de fond d'écran : manquait productOptions\productPolicy et diverses clés
- snippingtools ne fonctionnait pas : manquait productOptions\productPolicy
- Clic droit sur le fond d'écran « display, personnalization » ne lance rien : nécessite le sous système 32 bits
- Mstsc : OK avec NLA,
- BITS et Powershell : OK dans une session adm

Anomalies apparues avec la version 1607 build 14393

- Lors de l'ouverture de session « adm », un message d'erreur de logonui.exe apparaît mais n'est pas bloquant
- WMP ne lit pas les fichiers .wav, mais il lit les fichiers MP3
- Dans le cas d'un VHD, la session Adm ne s'ouvre pas (contournement possible)

Anomalies persistantes

- Dans le cas du démarrage sur une clé USB, et lors de l'ouverture de la session « adm », le lancement d'un programme génère un warning « Editeur non vérifié ». Cette anomalie n'apparaît pas dans une VM avec un VHD contenant une image « flat » de boot.wim.
- premier lancement de Powershell très long : pb de .cat
Si je copie tous les fichiers cat de install.wim alors il faut attendre une minute avant que powershell devienne opérationnel. Avec Procmon, on constate qu'un svchost crée le fichier « ...\cartoot2\...\catdb ».
- impossible d'épingler dans la barre de tâches avec un clic droit

- l'icône « éjection USB » ne permet pas l'éjection car il manque une entrée dans le menu contextuel. Ce n'est pas très grave car il n'y a pas de cache en écriture sur les clés USB.
- remote powershell et Wsman?
- trace réseau avec netsh : ndiscap.sys démarre, le fichier ETL est généré mais pas le fichier CAB
- ...

DCOM et la bascule vers l'interface graphique classique

La nouvelle interface graphique appelée « Metro » semble impossible à mettre en œuvre dans Winpe. Néanmoins certaines actions sont définies par la valeur d'un pointeur dans le registre. Il est parfois possible de modifier ce pointeur et d'activer l'interface classique d'un composant.

Une comparaison avec windows7 permet d'identifier les divergences relatives à la nouvelle interface.

Par exemple, le clic droit sur le bureau permet de sélectionner les éléments « Paramètres d'affichage » ou « Personnaliser » du panneau de configuration. Ceci déclenche l'affichage de la nouvelle interface. Une modification judicieuse permet de retrouver l'ancienne interface.

- Nouvelle interface 'metro' :

```
HKCR\DesktopBackground\Shell\Display\command\DelegateExecute={556FF0D6-A1EE-49E5-9FA4-90AE116AD744}
```

qui active l'objet COM : CLSID_LaunchSettingsPageHandler

- Retour à l'ancienne interface :

```
HKCR\DesktopBackground\Shell\Display\command\DelegateExecute={06622D85-6856-4460-8DE1-A81921B41C4B}
```

qui active l'objet COM :COpenControlPanel

Mais dans le cas de l'affichage, on est confronté au comportement décrit au chapitre suivant.

DCOM sur 64 bits : un bug avec Explorer ?

Dans Winpe 64bits, si on lance « Desk.cpl » par exemple, on constate dans le gestionnaire de tâches que le logiciel dllhost.exe est chargé. Ce programme est un « surrogate » faisant partie de l'architecture DCOM. Aucun affichage n'a lieu. Un nouveau lancement de « Desk.cpl » charge un nouveau processus « dllhost.exe ». Le dialogue client-serveur DCOM semble impossible.

Dans les mêmes conditions avec un Windows10 64bits, avec « Procmon », on constate le chargement de « x:\windows\syswow64\dllhost.dll » puis rapidement son déchargement. Il s'agit bien de la version 32 bits de 'dllhost'.

Or, dans la version classique de Winpe64 de l'ADK, le sous-système 32 bits est absent. D'où l'anomalie.

Est-il possible d'imposer à « Explorer » de charger un serveur DCOM 64 bits de cette fonctionnalité ?

Sites à consulter :

Securité dans COM : [https://msdn.microsoft.com/en-us/library/ms693319\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms693319(v=vs.85).aspx)

dllhost : [https://msdn.microsoft.com/en-us/library/ms695225\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms695225(v=vs.85).aspx)

APPID : [https://msdn.microsoft.com/en-us/library/ms678477\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms678477(v=vs.85).aspx)

<https://blogs.msdn.microsoft.com/jigarme/2007/10/09/what-is-appid/>

Com-32-64 : <http://mariusbancila.ro/blog/2010/11/04/32-bit-and-64-bit-com-servers/>

Wow64 : le sous-système 32 bits dans Winpe

WinPeSe le propose. C'est donc possible.

Un site chinois (mais j'ai oublié lequel) explique que Winlogon crée normalement 2 objets system pour le fonctionnement du sous-système 32 bits.

Un développeur chinois a donc écrit le logiciel « SetWow64.exe ». Ce logiciel est repris par l'équipe de WinPeSe.

Le sous-système32 nécessite aussi d'enrichir le répertoire 'WinSXS'.

Que sont les objets system ?

Chercher sur MSDN et consulter les sites ci-dessous.

En français :

<http://www.ivanlef0u.tuxfamily.org/?p=39>

Object Directories :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff557755\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff557755(v=vs.85).aspx)

NtCreateDirectoryObject :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff556456\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff556456(v=vs.85).aspx)

ZwCreateDirectoryObject :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff566421\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff566421(v=vs.85).aspx)

Pour explorer les objets system : 'winobj.exe'

Il faut le télécharger depuis le site 'technet/sysinternals'. Il existe en version 32bits uniquement.

On ne peut donc pas l'utiliser avec un WinPe 64bits normal. C'est aussi pour combler cette lacune que j'ai écrit le script PS.

Le script 'MonSetWow64.PS1'

J'ai remplacé le programme « SetWow64.exe » par un script PS (Règle 1) 'MonSetWow64.PS1'. Il propose deux fonctionnalités :

- Lancé sans paramètre, il permet de visualiser les objets du système (comme winobj) dans un

système 64 bits.

- Avec le paramètre 'create', il crée les 2 objets nécessaires.

Il est utilisable même si l'on ne dispose pas du sous-système 32 bits. Cela faisait longtemps que je voulais un moyen de visualiser les objets du système en 'full 64 bits' mais je ne connaissais pas les API 'NTxxx, ZWxxx'.

Je sais que ce script nécessite encore du travail.

Avec Wow64 activé, le dialogue DCOM précédent (chapitre précédent) utilise le sous-système 32 bits car « Desk.cpl » fonctionne et la trace « procmon » confirme le lancement de la version 32 bits de « dllhost.dll ».

ProductPolicy : Wallpaper, SnippingTool.exe ...

La clé :

« HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\ProductOptions\ProductPolicy » contient les informations nécessaires pour activer certaines fonctionnalités ou composants comme le fond d'écran, le programme « snippingtool.exe », etc.

Son contenu change avec les versions de Windows et avec les composants installés.

Cette clé est exportée depuis un Windows10 build 10586 actif car je ne sais pas comment faire autrement.

Je n'aurais jamais trouvé ce point sans WINPESE. La comparaison a été longue.

Il y a un logiciel sur le net qui permet de visualiser. Il permet de modifier cette clé on mode offline.

<http://reboot.pro/topic/20585-productpolicy-viewer/>

<http://www.remkoweijnen.nl/blog/2010/06/15/having-fun-with-windows-licensing/>

Modification du wallpaper et de themecpl.dll

Le contexte

La première étape a consisté à rendre possible l'accès au menu « Personnaliser » ("Personalization"). Merci à ceux qui ont trouvé le mécanisme de changement de GUID dans la clé "...\DesktopBackground".

Ce menu est accessible à partir d'un clic droit sur le bureau .

On peut ouvrir la fenêtre « Personnaliser » ("Personalization") avec la commande :

```
control.exe /name Microsoft.Personalization
```

La fenêtre « Personnaliser » ("Personalization") offre en bas un lien "Desktop Background" qui permet de changer le wallpaper.

On peut ouvrir cette fenêtre avec la commande :

```
control.exe /name Microsoft.Personalization /page pageWallpaper
```

On peut aussi ouvrir la fenêtre « Couleur » ("Color") avec :

control.exe /name Microsoft.Personalization /page pageColorization

Sites :

une porte d'entrée mais il faut remonter dans l'arborescence du site :

[https://msdn.microsoft.com/en-us/library/bb773213\(VS.85\).aspx](https://msdn.microsoft.com/en-us/library/bb773213(VS.85).aspx)

modification de shell32.dll :

<http://superuser.com/questions/494878/does-windows-8-have-a-status-bar-to-display-details-of-a-file>

idem

<http://sumtips.com/2012/11/windows-8-move-details-pane-to-bottom-of-explorer-window.html>

divers

<http://vistastylebuilder.com/forum/index.php?topic=215.0>

Premier écueil

Mais si on tente d'ouvrir cette fenêtre en cliquant sur le lien « Arrière plan du bureau » ("Desktop Background"), on obtient l'erreur « ms-setting:personalization-background : le module spécifié est introuvable » ("ms-setting:personalization-background : the module is not find"). En effet, ce lien pointe sur une application de l'interface Metro inexploitable dans WinPe.

Question : comment rendre actif ce lien ?

Après quelques recherches, en ouvrant le fichier themecpl.dll avec notepad, on trouve des chaînes de texte qui ressemblent à des scripts, à du xml. Elles contiennent les mots "duixml", "shellexecute", "pageWallpaper" et "pageColorization". En utilisant un logiciel explorant les ressources d'un programme, on retrouve les scripts dans les ressources. Une lecture permet de vite se faire une idée de la logique.

Le GUI de « Sons » ("Sounds") est lancé avec la commande :

"shellexecute="%windir%\system32\control.exe" shellexecuteparams="/name Microsoft.Sound /page 2"

Le GUI de l'écran de veille est lancé avec :

"shellexecute=%windir%\system32\rundll32.exe" shellexecuteparams="shell32.dll,Control_RunDLL desk.cpl,ScreenSaver,@ScreenSaver"

On trouve vite comment modifier le texte pour lancer les deux commandes qui nous intéresse.

La seule ressource à modifier : « UIFILE/1001 »

Les deux chaînes de texte à modifier :

Avant	shellexecute="ms-settings:personalization-background"
Après	"shellexecute="%windir%\system32\control.exe" shellexecuteparams=" /name Microsoft.Personalization /page pageWallpaper"

Et

Avant	shellexecute="ms-settings:personalization-colors"
Après	"shellexecute="%windir%\system32\control.exe" shellexecuteparams=" /name Microsoft.Personalization /page pageColorization"

L'outil pour manipuler les ressources, explorer et modifier (gros bouton vert de compilation) et sauvegarder les changements : ResourcesHacker ici <http://www.angusj.com/resourcehacker/>

Il suffit de remplacer le fichier d'origine dans WinPe(le fichier mui est inchangé).

Ensuite on teste ce nouveau fichier dans WinPe.

Second écueil

Le second écueil est plus difficile à décrire sans copie d'écran. Mais cela allongerait trop le PDF.

La combobox "picture location" contient plusieurs entrées. Elles sont inopérantes dans mon contexte.

Seule "Solid Colors" est active. Je ne sais pas pourquoi.

Une solution :

- Lors de la construction de winpe, je dépose les images "wallpaper" dans un répertoire à la racine de X, soit x:\fleurs dans mon cas. Comme j'utilise une VM, cela devient très simple. Pour un test sans VM, il faut monter le fichier Boot.wim avec Dism par exemple, et copier le répertoire à la racine. Puis démonter.
- Dans le bureau de WinPe, j'ouvre le menu «Desktop Background » (faire un clic droit sur le fond d'écran et activer le lien "Desktop Background" ou lancer la bonne commande control, voir ci-dessus).
- Cliquer sur le bouton « Parcourir... » ("browse...").
- Sélectionner le répertoire contenant les images. Soit x:\fleurs dans mon cas.
- Dans la combobox, sélectionner l'entrée correspondant au répertoire. Et les images apparaissent.

Vous pouvez alors changer le fond d'écran directement.


J'espère que quelqu'un va trouver pourquoi le fond d'écran initial n'apparaît pas, ni même son entrée correspondante dans la combobox.


Modification impossible en PS ? Est il est possible de compiler des ressources en C#, avec csc par exemple ?

IE Full64 : presque complet

Info : IE nécessite les clés :

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings

 Si la valeur hku\.default\...\cache\persistent (dw 1) est absente, alors IE lance 3 rundll32 de suppression des traces

 A noter la présence d'un programme de test :

"X:\Program Files\Internet Explorer\iediagcmd.exe".

Il utilise Dxdiag, nesth, ipconfig... Le fichier log produit me semble complet.

clé x86AppPath

J'ai fait trois tests :

- Premier test : Le sous-système 32 bits est actif
 - Lancement de iexplore 64 bits : affichage fugitif
 - Recopie du répertoire x86 de internet explorer : une fenêtre s'affiche et procmon signale que iexplore 32 bits redémarre sans arrêt.
 - Lancement de iexplore 32 bits : affichage message d'erreur « 0xc0000034 démarrage impossible ».
- Second test : sous-système 32 bits inactif mais le répertoire x86 de internet explorer est présent
 - Lancement de iexplore 64 bits : la version 64 bits essaie de lancer la version 32 bits. Affichage d'une fenêtre.
- Troisième test : je lance iexplore 32 bits : affichage d'une fenêtre mais curseur « activité en cours »

Première recherche avec procmon :

iexplore 64 bits consulte la clé hklm\software\microsoft\internet explorer\main\

x86AppPath = x:\program files (X86)\internet explorer\iexplore.exe

Puis « createProcess » de cette version 32 bits avec les paramètres

« scodef:2760 credat:xxxx /prefech:2 » où scodef fournit le Pid du process iexplore 64 bits.

- Nouveaux tests :
 - supprimer cette valeur : affichage fugitif
 - faire pointer « x86AppPath » vers la version 64 bits : on arrive à voir plusieurs « IE » dans la barre de tâche et visualiser le contenu de « MSN ».

Utilité de Loosely-Coupled IE (LCIE) ?

Première découverte sur internet

<http://blog.httpwatch.com/2009/04/07/seven-things-you-should-known-about-ie-8/>

"TabProcGrowth=0 is a registry edit that will disable the Loosely-Coupled IE (LCIE) function in Internet Explorer 8. Essentially what this means is that all tabs in Internet Explorer will be handled by

one process of iexplore.exe. This also means that Protected Mode will be Off and that if one of your tabs crash, Internet Explorer will crash."

Il n'y a plus d'instances de « iexplore » avec le paramètre « scodef ». Il faut utiliser la barre de tâche et les icones de IE pour passer d'un onglet à l'autre.

La fenêtre principale de IE contenant les menus s'affiche !

J'ai fait le test avec le build 14393. Le hasard une fois de plus...Il manquait simplement le fichier IeFrame.dll.Mui.

F12 (debugger IE)

Pour l'instant il n'est pas entièrement opérationnel. La trace réseau fonctionne.

Anomalies identifiées

Session « System » :

- Divers menus ne fonctionnent pas
- le téléchargement : les fichiers sont bien téléchargés dans un répertoire local mais ne sont pas affichés
- Qwant : la page d'accueil se charge mais la recherche n'est pas lancée !

Session « ADM » :

- La gestion des mots de passe ne fonctionne pas
- Les téléchargements fonctionnent ainsi que l'affichage des téléchargements



Si la date et l'heure de Winpe n'est pas correcte, alors certains sites rejettent la demande d'authentification.

Powershell : démarrage très lent

Le premier démarrage de Powershell est toujours très long.

Dans le forum <http://theoven.org/index.php?topic=1639> : « sezz » explique :

I also figured out why starting PowerShell in Windows PE is extremely slow (first start takes about 20 seconds for me):

- NGEN hasn't run yet -> the native image cache doesn't exist
- CATDB hasn't been generated yet

Running "NGEN.EXE update /NoDependencies /Silent" takes ages, so I ran it once and fetched the files from "X:\Windows\assembly", now I theoretically could use them everytime when building an image, but they are very large (about 300MB)...

The CATDB gets created on the first start of PowerShell.exe, that's why it takes so long. The file "X:\WINDOWS\SYSTEM32\catroot2\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\catdb" can

be backed up (stop CryptSVC service first) and then used when building an image. It's only about 20MB.

Couldn't find a solution yet without running Win PE first and grabbing the CATDB, but it's good enough for now :)

Changement du nom du « computer »

Il existe 2 noms, un pour la machine et un pour la réseau.

Winpe génère normalement un nom aléatoire pour le « computernome ».

Le script de WinPeSe indique qu'il faut modifier la clé suivante pour imposer un nom prédéfini :

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinPe
    SetComputerName = 0 type Reg_DWORD
```

La question est : comment a été identifié cette clé ?

Faire le test sans cette clé !!!

Puis le nom du « computer » doit être inscrit dans les clés suivantes :

```
...\SYSTEM\ControlSet001\Control\ComputerName\ActiveComputerName
    ComputerName = « MyComputerName »
...\SYSTEM\ControlSet001\Control\ComputerName\ComputerName
    ComputerName = « MyComputerName »
...\SYSTEM\ControlSet001\Control\services\TcpIp\Parameters
    HostName = « MyComputerName »
...\SYSTEM\ControlSet001\Control\services\TcpIp\Parameters
    « NV HostName » = « MyComputerName »
```

L'ouverture de session avec le compte « administrateur »

Sans le travail de l'équipe WinPeSe, je n'aurais jamais su mettre en œuvre l'ouverture de session avec le compte « administrateur ».

Les sessions natives dans WinPe :

- Session 0 : pour les services
- Session 1 : automatiquement créée par WinPe lors de son démarrage pour le compte « system ».
- Session 2 : elle sera créée lors de l'ouverture de la session par le compte "adm"

Dans cette session « adm », le compte « administrateur » a tous les droits puisqu'il appartient au

groupe des administrateurs. Mais ces droits ne sont pas actifs !!!

Le principe général lu dans WinPeSe

Lors de son démarrage, Winpe active la session pour l'utilisateur "system".

On prépare les clés de l'ouverture automatique d'une session : « mécanisme AutoAdminLogon ».

Il faut absolument arrêter et modifier la configuration « start= disabled » des deux services Gpsvc et TrustedInstaller.

Depuis la session "system", on déconnecte la console de cette session avec "tsdiscon.exe"

La session "system" n'est pas fermée. Mais la console "clavier/écran/ressources" n'est plus affectée à cette session.

Winpe comprend que la console "clavier/écran/ressources" redevient disponible. Il propose donc à un autre utilisateur d'ouvrir une session avec le GUI de « logon ».

Or, le mécanisme « AutoAdminLogon » a été préalablement configuré. Donc Winpe dispose des éléments d'authentification du nouvel utilisateur (workgroup, compte, mot de passe)

Le mécanisme d'ouverture de session contrôle l'identité du compte « adm ». Puis crée le répertoire de profil pour cet utilisateur.

On dispose donc de deux sessions interactives, l'une pour le compte "system" et l'autre pour le compte "administrateur".

On passe de l'une à l'autre avec la commande "tscon", « tscon 1 » pour activer la session "system" et "tscon 2" pour activer la session "administrateur".



Pour éviter le piège du mot de passe vide, je change le mot de passe de l'administrateur. Modifier le script et le mot de passe en fonction de votre besoin.

Le « computer » doit appartenir à un « workgroup »

Il faut s'assurer que le « computer » a bien rejoint un « workgroup » pour ne pas avoir ultérieurement le message d'erreur "domaine ou workgroup inaccessible".

Si l'on force la valeur des 4 clés définissant le nom du computer, alors le computer n'est pas enregistré automatiquement dans le « worgroup ».

Pour rejoindre un "workgroup", on peut utiliser l'API "joinDomainOrWorgroup" de WMI par exemple.



Script en vbs pour tester dans WinPeSe.

```
strComputer = "."
Set oWmiService = GetObject("winmgmts:\\.\root\CimV2")
Set colComps = oWmiService.ExecQuery ( "Select * from Win32_ComputerSystem" )
For Each oComp in colComps
  wscript.echo "oComp.workgroup"
  res = oComp.JoinDomainOrWorkgroup("Workgroup")
```

```
wscript.echo res  
Next
```

Script en Ps pour MicroWinpe.

```
$s = gwmi win32_computersystem  
$.JoinDomainOrWorkgroup('WorkgroupWinpe')
```

 Surprise :

Avec un clic droit sur "propriétés" de "Ce PC", on constate :

"Groupe de travail : non disponible"

Puis si on clic "Modifier les Paramètres" puis "Computer Name", on constate :

"workgroup = *Unknown*"

? Point à vérifier :

Si on ne fixe pas la valeur des 4 clés définissant le nom du computer, il semble que "wpeinit" assure automatiquement l'intégration du computer dans le « workgroup ».

Ralentissement à l'ouverture : corrigé

Il existe de nombreuses sources de ralentissement de l'ouverture de session 'administrateur'. J'ai identifié celles-ci :

- affichage du message « Patientez ... » pendant 10 minutes

Cela se produit si les deux services GpSvc et/ou TrustedInstaller ne sont pas configurés avec « start = demand »

- le réseau est monté avant l'ouverture de session

C'est wpeinit qui monte le réseau. Avec procmon, je constate que winlogon tente d'utiliser la ressource :

\\LsaSetupDomain*\MAILSLOT\NET\NETLOGON

Un time-out d'environ 6 secondes semble actif. Et l'appel est relancé plusieurs fois.

J'ai réalisé deux tests :

- retrait de la carte réseau de la VM
- présence de la carte mais retrait du lancement du programme wpeinit

Dans les deux cas : les écritures ne se font pas car le mailslot est « disconnected » mais il n'y a pas de ralentissement.

- ntuser.dat ne contient pas "UserSignedIn"


L'absence de la clé « hkcu\...\explorer\UserSignedIn » introduit un délai de presque une minute après

l'affichage de « Préparation... ».

- le service 'SENS' que j'ai mis en place pour BITS introduisait un autre retard de deux minutes. Winlogon envoie des notifications à divers programmes et attend leur réponse. Dans mon cas, le service 'SENS' ne fonctionnait pas correctement.

Les notifications de Winlogon sont visibles dans cette clé qu'il faut modifier :

```
"...\ControlSet001\Control\Winlogon\Notifications\Components\"
```

 Surprise : « LsaSetupDomain » est aussi visible dans les trames SMB envoyées sur le réseau lors d'une connexion à une imprimante par exemple.

Erreur 0x80000003 : plusieurs cas de figure depuis le build 14393

Dans le cas de l'ouverture de la session "administrateur", il apparaît l'erreur 0x80000003. Elle n'est pas bloquante. Il suffit de cliquer sur le bouton « OK » et l'ouverture de session a bien lieu.

Dans le cas du « switch user », la tentative d'ouverture d'une nouvelle session bloque le système.

Dans le cas d'un démarrage depuis un VHD en mode "flat" (depuis une VM hyperV ou depuis un disque dus USB), la session "administrateur" ne s'ouvre pas.

Le hasard a bien fait les choses : un site proposait en téléchargement un fichier de 4Go contenant une version de Winpese et un ensemble d'applications installées (principalement des applications portables ne nécessitant pas de recherches pour leur intégration). Voir <http://theoven.org/index.php?topic=1873.0>. Et là, tout fonctionne. Ce qui a permis de trouver une solution efficace.

Solution :

Le fichier **Imageres.dll** de l'ADK est plus petit que celui de la référence ISO.

Il manquait aussi le fichier « InputSwitch.dll ».

Autre contournement possible (pour mémoire):

renommer Windows.UI.Logon.dll en Windows.UI.Logon.dl_

Dans l'état actuel des scripts, il est possible de créer un fichier x:\NoAdm qui interdit le lancement de "tsdiscon.exe". On peut donc faire toutes les investigations nécessaires.

Désactivation de la partie graphique de logonUI

Il est possible de désactiver le parti graphique de « logonUI ». Il suffit de renommer le fichier « windows.UI.logon.dll » (en « -windows.Ui.logon.dll » par exemple).

La partie logonUI en mode console (« ConsoleLogon.dll ») affiche une boîte noire et montre les mêmes textes mais sans les animations.

Une autre possibilité : remplacer logonUI.exe par cmd.exe

Cela m'a permis de tracer avec « procmon », bien que sans résultat. Mais la méthode peut me servir un jour, peut être:

- Renommer logonUi.exe en -logonUi.exe
- Copier cmd.exe en logonUi.cmd
- Préparer les clés et les commandes nécessaires pour l'ouverture de session avec administrateur
- Lancer procmon
- Lancer la procédure d'ouverture de session : tsdiscon
- CMD prend la main. Lancer un powershell. Récupérer avec gwmi la ligne de commande de CMD. Lancer alors -logonUI.exe avec les paramètres de la ligne « cmd ».

Lors d'échec, -logonUI boucle. On peut faire un "CTRL-C" pour en sortir.

On retourne donc au CMD. Et avec ALT-TAB, on revient à powershell. On peut alors retourner dans la session "System" avec tscon 1. Et arrêter Procmon.

Runas uniquement depuis la session ADM

On peut vérifier le bon fonctionnement ainsi :

```
net user MonToto MyPassword /add
net localgroup administrateurs MonToto /add
runas /noprofile /user:MonToto cmd.exe
puis taper le mot de passe « MyPassword ».
```

Ne pas oublier le paramètre « /noprofile » sinon il y a l'erreur « le périphérique n'est pas prêt ».

La commande Runas ne fonctionne pas depuis la session System.

Le warning « L'éditeur n'a pas pu être vérifié »

Lorsque Winpe démarre depuis un clé USB ou dans une VM avec un fichier iso, tous les programmes lancés par « Windows+R » déclenchent l'affichage de ce warning. Mais si je décompresse le fichier boot.wim dans un VHD (avec un BCD adapté), ce warning n'apparaît pas.

Je constate que le seul programme qui ne déclenche pas ce warning est : tscon.exe

Contournement : lors de l'apparition de ce warning, basculer dans la session « system » avec « tscon.exe 1 ». Puis revenir dans la session « adm » avec « tscon 2 ». Le programme powershell en cours et ses descendants déclencheront encore le warning mais pas les nouveaux programmes.

Surprise : avec la fonctionnalité « Open Command Here », le lancement du programme cmd.exe ne déclenche pas ce warning.

Site décrivant la fonctionnalité « Open Command Here » (voir option 8 dans la page) :

<http://www.tenforums.com/tutorials/3288-command-prompt-open-windows-10-a.html>

Pour une mise en œuvre dans le menu contextuel sans utiliser la touche Shift:

<http://www.askvg.com/enable-open-command-window-here-option-in-context-menu-in-windows-vista>

Supprimer les valeurs :

HKEY_CLASSES_ROOT\Directory\Background\shell\cmd\extended

HKEY_CLASSES_ROOT\Directory\shell\cmd\extended

La trace réseau avec netsh

[https://technet.microsoft.com/en-us/library/dd878517\(v=ws.10\).aspx#bkmk_traceStart](https://technet.microsoft.com/en-us/library/dd878517(v=ws.10).aspx#bkmk_traceStart)

Actuellement :

- elle reste incomplète
- le service PLA est obligatoire pour un «assez bon » fonctionnement de la commande « stop ».
ce service PLA nécessite la modification des ACL de la clé « service\pla\configuration »
- la commande « stop » ne génère pas l'ensemble des fichiers attendus (pas de fichier .cab...)
- le fichier ETL se remplit uniquement si le pare-feu autorise les « inbound connections » ou si une exception est créée

La collecte des informations system nécessite le service « Schedule ». Mais il bloque la recopie des fichiers dans une VM. Donc je ne démarre pas.

Beaucoup de recherche pour un petit gain, le log du pare-feu contient un meilleur résumé des trames du réseau.

L'utilisation des providers donne actuellement de meilleurs résultats que l'utilisation des scénarios.

L'installation du pilote ndiscap.sys

Fichiers utilisés : ndiscap.inf, ndiscap.sys et le .cat identifié avec « signtool ».

Le fichier « .cat » est il vraiment utile ?

L'installation du pilote dans la pile « network »est réalisée après le démarrage de « Winpe » :

```
netcfg -c p -i MS_NDISCAP
```

Le paramètre « -e » de netcfg.exe interdit l'installation du pilote. Incompréhensible !

Elle nécessite le fichier « ndiscapfcg.dll ».

Le démarrage réussit : net start ndiscap

La capture est possible avec une carte Wifi ou Ethernet. Ok dans une VM.

La configuration avec netsh

Il faut ajouter dans la ruche « system » les clés « netdiagfx » et « Nettrace » qui contiennent les scénarios disponibles.

Ces deux clés sont protégées par des ACL.

- Pour démarrer une capture avec un provider :

```
netsh trace start provider=Microsoft-Windows-TCPIP capture=yes report=yes correlation=yes overwrite=yes level=5
```

- Pour démarrer une capture avec un scénario :

```
netsh trace start scenario=InternetClient capture=yes report=yes
```

- Pour arrêter une capture :

```
netsh trace stop
```

La commande « netsh trace stop » ne génère pas le fichier « .Cab » ni le fichier « report ».

- Pour convertir le fichier ETL en TXT :

```
netsh trace convert input=<chemin>\NetTrace.etl dump=TXT
```

Le fichier de type « txt » ne contient pas toutes les informations attendues mais le trafic TCP est présent.

Voir <https://isc.sans.edu//diary/No+Wireshark?%2bNo%2bTCPDump%3f%2bNo%2bProblem!/19409>

Voir absolument : <https://www.microsoft.com/en-us/download/details.aspx?id=44226>

Le journal d'événement

Ne pas oublier d'enrichir les clés '...\services\eventlog\applications' et '...\services\eventlog\system'.

'Eventlog.msc' affiche bien les journaux. Pour cela, il faut mettre en œuvre la séquence suivante :

- arrêter le service eventlog
- renommer la clé MiniNt
- redémarrer le service eventlog

Bizarrement, il arrive que les événements ne soient plus inscrits.

Les audits

L'activation de certains audits permet d'enrichir le journal d'événements « Security ».

```
auditpol /list /category /v
```

Category/Subcategory	GUID
Accès aux objets	{6997984A-797A-11D9-BED3-505054503030}

Accès DS	{6997984F-797A-11D9-BED3-505054503030}
Changement de stratégie	{6997984D-797A-11D9-BED3-505054503030}
Connexion de compte	{69979850-797A-11D9-BED3-505054503030}
Gestion des comptes	{6997984E-797A-11D9-BED3-505054503030}
Ouverture/Fermeture de session	{69979849-797A-11D9-BED3-505054503030}
Suivi détaillé	{6997984C-797A-11D9-BED3-505054503030}
Système	{69979848-797A-11D9-BED3-505054503030}
Utilisation de privilège	{6997984B-797A-11D9-BED3-505054503030}

Pour connaître les événements de sécurité tracés actuellement :

```
auditpol.exe /get /category:*
```

Les deux commandes à lancer :

```
'auditpol.exe /set /Category:{6997984C-797A-11D9-BED3-505054503030} /success:enable /failure:enable'
```

```
'auditpol.exe /set /Category:{69979848-797A-11D9-BED3-505054503030} /success:enable /failure:enable'
```

La commande qui génère un BSOD de Winpe :

```
'auditpol.exe /set /Category:{6997984A-797A-11D9-BED3-505054503030} /success:enable /failure:enable'
```

Le firewall : ses logs, sa configuration, les profils

Quelques commandes utilisées par gatherNetworkInfo.vbs pour connaître l'état actuel du pare-feu :

```
netsh advfirewall monitor show currentprofile
```

```
netsh advfirewall monitor show firewall
```

```
netsh advfirewall monitor show consec
```

```
netsh advfirewall firewall show rule name=all verbose
```

```
netsh advfirewall consec show rule name=all verbose
```

```
netsh advfirewall monitor show firewall rule name=all verbose
```

```
netsh advfirewall monitor show consec rule name=all
```

Les logs

Avec l'audit, le journal des événements enregistre l'activité du firewall. Néanmoins, il est possible

d'obtenir un fichier de log plus spécialisé. Pour cela, lancer « WF.msc ». Puis, cliquer sur « propriétés » (un petit lien sous le dernier profil), et activer les log.

☛ Attention : les écritures dans le fichier de log sont en retard d'au moins 30 secondes sur l'événement. Donc, patience !

Activation/désactivation

Il est préférable de laisser le service actif et d'autoriser tous les flux. Nombreuses applications interrogent le « Firewall » et ne fonctionnent pas si ses API ne répondent pas. Même point d'entrée pour la configuration que ci-dessus.

Par exemple, si on arrête le service pare-feu avec « net stop MpsSvc », alors les « ping » vers WinPe échouent.. Mais si on démarre le pare-feu avec « net start MpsSvc » et que l'on autorise les « inbound connections » avec « Wf.msc », alors les « ping » vers WinPe réussissent.

Utile avec les WinPe sans MMC : Wpeutil disablefirewall ou Wpeutil enablefirewall

Comment changer le profile du réseau de public en privé ? NOT OK !

Information trouvée sur internet mettant en œuvre le service « NetProfM » (Network List Service) et son interface DCOM (voir le GUID de la commande):

```
# Get network connections

$networkListManager =
[Activator]::CreateInstance([Type]::GetTypeFromCLSID([Guid]"{DCB00C01-570F-4A9B-8D69-199FDBA5723B}"))

$connections = $networkListManager.GetNetworkConnections()

# Set network location to Private for all networks

$connections | % {$_ .GetNetwork().SetCategory(1)}
```

Le centre « Réseau et Partage » permet de constater le changement. Mais WF.MSC indique que le profile est toujours public : ce changement est donc inopérant.

Autres sites

Avec GUI de « home folder »

<https://tinkertry.com/how-to-change-windows-10-network-type-from-public-to-private#Fix2WiFi>

Avec le registre

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles

puis dans les sous-clés, la valeur « Category » décrit le type de profil :

Public: (leave this blank)

Private: 1

Domain: 2

Avec Powershell

<http://www.tenforums.com/tutorials/6815-network-location-set-private-public-windows-10-a.html?ltr=N>

Get-NetConnectionProfile

Set-NetConnectionProfile -Name "xxxxxxxxxx" -NetworkCategory Private

De l'information sur nla et parefeu :

<https://blogs.technet.microsoft.com/networking/2010/09/08/network-location-awareness-nla-and-how-it-relates-to-windows-firewall-profiles/>

Le centre « Réseau et Partage » : avec build 14393 ?

Il faut installer le service « NetProfM ». Et pour rendre ce service opérationnel, il faut neutraliser temporairement la clé « SystemSetupInProgress » pendant son démarrage.

Pas vraiment utile !

WinRm dans la session adm : OK dans les deux sens

Le service WinRm(nommé précédemment WsMan dans windows 8) permet d'envoyer des commandes PS vers une machine distante. Il permet aussi d'en recevoir.

Il utilise deux ports, l'un (47001) pour le configurer , l'autre (5985) pour recevoir les commandes lancées depuis un ordinateur distant.

Après son démarrage, on peut vérifier qu'il a bien ouvert le port TCP 47001 avec la commande netstat permettant d'obtenir le pid (ou le nom avec -anb) et une consultation en boucle si besoin !

```
NetStat -ano
TCP 0.0.0.0: 47001    0.0.0.0:0    LISTENING    4
TCP 0.0.0.0:5985    0.0.0.0:0    LISTENING    4
```

WinRm nécessite une étape de configuration avec la commande « Winrm quickconfig -q ». Puis, hors domaine, il faut déclarer les machines de confiance.

```
Winrm Quickconfig -q
Winrm Set winrm/config/client '@{TrustedHosts="*"}'
```

Mais en l'état, ces commandes échouent.

Anomalies constatées : divers messages « access denied »

Que faire ? changer le compte du service WinRm en 'LocalSystem' ou modifier le contenu des groupes locaux ainsi :

```
net localgroup WinRMRemoteWMIUsers__ /add
net localgroup administrateurs system /add
net localgroup administrateurs localservice /add
net localgroup administrateurs networkservice /add
```

Existe t-il un compte « winrm » dans les ACL ?

J'ai toujours la même erreur « access denied ». Et puis un coup de chance qui relève de la sérendipité (une question du jeu des mille euros en france).

Découvert par hasard, avec le démarrage du service « Serveur de fichiers », on avance un peu et on obtient l'erreur déjà rencontrée dans winpe4 ou 5 pour Windows 8.1.

```
Net start lanmanserver
```

Je change aussi le mot de passe du compte « administrateur » pour pouvoir établir des connexions distantes « entrantes ». Je partage une ressource pour d'éventuelles copies de fichiers. Et j'installe le service « SecLogon ».

```
net user administrateur +Noel
net share monx=x:\
```

Le nouveau message pour « Winrm QuickConfig -q » devient :

```
WSManFault
  Message
    ProviderFault
      WSMANFault
        Message = WinRM firewall exception will not work since one of the network connection
types on this machine is set to Public. Change the network connection type to either Domain or
Private and try again.
```

Il s'agit d'un warning indiquant que le port TCP 5985 n'est pas ouvert pour le trafic entrant WinRm.

La configuration des machines de confiance réussit :

```
Winrm Set winrm/config/client '@{TrustedHosts="*"}'
```

➤ Pour les commandes sortantes WinRm

Powershell fait un contrôle de l'environnement Winpe et signale que les accès distants avec WinRm ne fonctionnent pas dans Winpe. On peut neutraliser temporairement la clé MiniNt et lancer Powershell. Et ainsi les commandes WinRm sortantes réussissent.

Pour neutraliser temporairement la clé MiniNt :

```
reg DELETE HKLM\system\currentcontrolset\control\miniNt /f
start-process PowerShell -argumentList '-executionpolicy unrestricted'
start-sleep -s 1
reg ADD HKLM\system\currentcontrolset\control\miniNt /f
```

Pour vérifier que les commandes sortantes WinRm fonctionnent :

```
$s="192.168.0.12" # my win10 computer
# ask the password for the remote computer
$c=get-credential WIN-LBH1HBGLMAA\noel
invoke-command -computername $s -credential $c -scriptblock {$env:computername}
invoke-command -computername $s -credential $c -scriptblock {gwmi win32_bios}
```

➤ Pour les commandes entrantes WinRm

J'ouvre le port avec netsh.

```
netsh advfirewall firewall Add rule name="NONO-5985-winrm" dir=in protocol=tcp localport=5985 action=allow
```



Pour ouvrir le port, il faut que le service parefeu MpsSvc soit démarré. Il semble que ce soit « wpeinit.exe » qui le démarre. Et avec la session Adm le script ne lance plus cette commande au bon moment. Je modifie mes scripts PS pour ouvrir ce port.

Pour les tests, je lance manuellement les commandes suivantes depuis une console :

```
net stop mpssvc
winrm quickconfig
net start mpssvc
```

Et ainsi le port 5985 est bien « listening » dans « netstat -ano ».

La commande « test-wsman <ip de winpe> » réussit quand elle est lancée depuis un pc windows10 distant.

Question : faut il vraiment passer de « public » à « privé » ?

<http://www.tenforums.com/tutorials/6815-network-location-set-private-public-windows-10-a.html?ltr=N>

Depuis un pc distant, je lance les commandes suivantes vers la machine « winpe »:

```
$s="192.168.0.15" # my winpe computer
# ask the password for the winpe computer
$c=get-credential MINWINPC\administrateur
invoke-command -computername $s -credential $c -scriptblock {$env:computername}
```

J'obtiens le message « [192.168.0.15] La connexion au serveur distant 192.168.0.15 a échoué avec le message d'erreur suivant: Le service WSMAN n'a pas pu lancer un processus hôte pour traiter la demande donnée. Assurez-vous que le serveur hôte du fournisseur et le proxy WSMAN sont correctement inscrits. »

La trace Procmon montre que le service Winrm ne lance pas le programme « wsmprovhost.exe ».

Je modifie la clé suivante dans la machine Winpe :

```
hklm\system\setup\SystemSetupInProgress = 0
```

Puis je relance la commande « invoke-command » depuis la machine Windows10.

Et la commande réussit !

BITS : opérationnel seulement avec la session Adm

Le service BITS nécessite des 'assemblies' et très peu de fichiers (qmgr.dll principalement).

Il est opérationnel dans la session administrateur mais non dans la session « system ».

La commande « Import-Module BitsTransfer » n'est pas utile dans mon contexte.

- La commande réussit dans la session Adm
- L'anomalie avec la session « system » :

```
net start bits
Start-BitsTransfer "http://noel.blanc.free.fr/index.php" "x:\"
```

Lors du lancement de 'Start-BitsTransfer, on obtient l'erreur suivante:

```
Start-BitsTransfer : L'opération demandée n'a pas été effectuée car l'utilisateur n'est pas
connecté au réseau. Le service spécifié n'existe pas. (Exception from HRESULT: 0x800704DD)
```

Idem avec bitsadmin.exe :

```
bitsadmin.exe /TRANSFER "toto" /DOWNLOAD "http://noel.blanc.free.fr/index.php" "x:\"
Unable to add file - 0x800704dd
L'opération demandée n'a pas été effectuée car l'utilisateur n'est pas connecté au réseau.
```

Ma seule piste pour l'instant est de chercher des informations sur cette erreur.

- Une page de Msdn fournissant la liste des erreurs :

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms681381\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms681381(v=vs.85).aspx)

Usually this happens when the task is configured to run only when the user is logged on.

- Une autre page de Msdn :

[http://technet.microsoft.com/en-us/library/cc720473\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc720473(v=ws.10).aspx)

```
ERROR_NOT_LOGGED_ON :
```

```
The SENS service is not receiving user logon notifications. BITS (version 2.0 and up) depends
on logon notifications from Service Control Manager, which in turn depends on the SENS service.
Ensure that the SENS service is started and running correctly.
```

D'où l'installation du service « SENS » et du service dont il dépend « EventSystem ».

Mais l'erreur est toujours présente dans la session « system » !

MSTSC et TermService

- Sens « sortant » de Winpe :

Le message d'erreur est le suivant : « The remote computer requires Network Level Authentication which your computer does not support. ».

Peut être la clé « LSA \LmCompatibilityLevel » ? Non, ni avec 2, ni avec 3.

- Sens « entant » dans Winpe : « connexion impossible » même après avoir autorisé les « inbound connections » dans le pare-feu.

Le service « TermService » signale qu'il démarre. Mais dans le journal d'événement « Windows/TerminalServices-localSessionManager », un message indique « les services du bureau à distance n'acceptent pas d'ouverture de session car le programme d'installation est en cours d'exécution ».

Le service « TermService » n'écoute pas sur le port 3389. Constat fait avec « netstat -ano ».

MSTSC from WinPe to Windows10 : c'est possible

Site :

<https://technet.microsoft.com/en-us/library/cc732713.aspx>

Network Level Authentication is an authentication method that can be used to enhance RD Session Host server security by requiring that the user be authenticated to the RD Session Host server before a session is created.

To determine whether a computer is running a version of Remote Desktop Connection that **supports Network Level Authentication**, start Remote Desktop Connection, click the icon in the upper-left corner of the Remote Desktop Connection dialog box, and then click About. Look for the phrase Network Level Authentication supported in the About Remote Desktop Connection dialog box.

<http://vidmar.net/weblog/archive/2007/05/27/The-remote-computer-requires-Network-Level-Authentication-which-your-computer.aspx>

Vérification préalable de la configuration MSTSC sur WinPe

- lancer MSTSC
- clic gauche sur l'icône dans le coin supérieur gauche ou clic droit dans la barre de titre
- menu « A propos »

Une fenêtre s'ouvre précisant si NLA est prise en charge. Par défaut, NLA n'est pas prise en charge dans WinPe.

Désactivation NLA sur la machine Windows10

Il y a au moins une configuration qui fonctionne : désactiver l'authentification NLA sur la machine windows10 distante.

Sur la machine Windows10, à partir du menu « propriétés système\utilisation à distance », il faut décocher la case « N'autoriser que la connexion des ordinateurs exécutant le bureau à distance avec l'authentification NLA ».

On peut ainsi établir la connexion avec la machine windows10 distante.

Activation de NLA sur WinPe

Mais est il possible de configurer l'authentification NLA sur WinPe ? OUI !

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig

Security Packages Reg_Multi_Sz kerberos, msv1_0 , tspkg, pku2u, cloudAP, wdigest, schannel

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders
SecurityProviders = credssp.dll

new files : tspkg.dll, credssp.dll

Win10pCap and Wireshark hors boot.wim



Je fais le choix suivant : l'installation présentée ici est lancée après le démarrage de WinPe. Cela signifie que l'installation sera à refaire à chaque démarrage. C'est très rapide et on en a pas besoin tous les jours.

Mais il serait aussi possible, si les fichiers sont recopiés au bon endroit sur le média, de relancer le programme « installer.exe » qui installe le pilote « win10pcap » dans la ruche « system ». C'est ce que je fais dans ma VM.

On peut faire le choix d'installer ce pilote win10Pcap dans boot.wim, soit avec DISM, soit en modifiant la ruche « system » (par exemple en modifiant le script traitement.ps1).

➤ Mon contexte :

une VM utilise un fichier VHD contenant WinPe en mode "flat".

L'installation pourrait être différente avec un autre contexte, par exemple WinPe contenu dans une clé usb

➤ Versions testées :

win10pcap 10.2.0.5002 version lue dans le fichier inf

wireshark 2.0.2

➤ Sites :

www.wireshark.org

<http://www.win10pcap.org/>

Win10pCap

L'installation affiche rapidement un message d'erreur « le répertoire d'installation doit être sur un disque dur local ».

Deux méthodes pour contourner :

- modifier le MSI avec ORCA
- ou extraire les fichiers (sys, inf, cat, etc) du fichier MSI avec 7z.

Obtenir ORCA

Je l'ai trouvé dans le SDK de windows7 (64 bits) dont j'avais gardé une ISO.

Dans le fichier « E:\Setup\WinSDKTools_amd64\cab1.cab » on trouve divers outils dont orca, signtools, makecert....

On ouvre ce fichier « cab1.cab » avec 7z. Et on extrait le fichier « WinSDK_Orca_Msi_5E20C107_DAA3_4D49_AFAE_7FB2594F0CDC_amd64 ».

On ajoute l'extension « .msi » à la fin de ce nom. On vérifie avec 7z qu'il contient bien une « structure msi ». On peut maintenant installer ce produit « orca.msi ».

Méthode 1 : Modification et installation de win10pCap.Msi avec Orca

On crée une copie de sauvegarde du fichier « win10pCap.msi ».

On lance Orca et on ouvre le fichier « win10pCap.msi ». Dans la partie gauche « Tables », on sélectionne « ControlEvent ». Dans la partie droite, on sélectionne la ligne « installDirDlg » dont la colonne « Argument » contient « VerifyReadyDlg ». On double-clic sur le champs de la colonne « Condition » de cette ligne. On peut maintenant modifier la valeur. On remplace tout le contenu « WIXUI_DONTVALIDATE...._VALID='1' » par « 1 ». On valide. On sauvegarde.

Ensuite on installe ce fichier Msi dans WinPe (double-clic, ou ouvrir, ou ...)

Méthode 2 : Extraction et installation du pilote win10pCap

Ouvrir le fichier « win10pCap.msi » avec 7Z.

Recherche à l'intérieur le fichier «win10pCap.cab ». Ouvrir ce fichier. Il contient les versions pour divers OS (windows 7/8 et windows 10, en version 32 et 64 bits). Extraire tous les fichiers dans un répertoire.

Et selon la configuration qui vous intéresse, copiez les trois fichiers (sys, inf, cat) dans un répertoire, ainsi que les deux fichiers wpcap.dll, paquet.dll, installer.exe.

Dans Winpe, on recrée l'arborescence suivante :

- « x:\program Files (x86)\win10pcap\x64\

- wpcap.dll, paquet.dll, installer.exe
- « x:\program Files (x86)\win10pcap\x64\drivers/win10 »
 - win10pcap.inf, win10pcap.sys, win10pcap.cat
- se placer dans le répertoire « x:\program Files (x86)\win10pcap\ »
- lancer « installer.exe ».

Wireshark

L'installation est simple : lancer le programme préalablement téléchargé depuis le site officiel.

Ne pas valider l'installation de WinPCap proposée par WireShark.

Il faut ajouter deux fichiers .dll suivants dans le répertoire x:\windows\system32 :

msvcp120.dll et msucr120.dll

Ces deux fichiers ne se trouvent pas dans la fourniture originelle de windows10Ent Build 10586. Je les prends sur ma machine sous windows10 hébergeant ma VM.

Ce qu'il faut savoir concernant l'installation de pilote

Pour installer un pilote, il faut généralement 3 fichiers.

- Un fichier .sys : c'est le code du pilote
- Un fichier .inf : il décrit les actions à réaliser lors de l'installation
- Un fichier .cat : c'est la signature des 2 fichiers précédents

Tous les pilotes sont signés dans la version 64bits (quid en 32 bits). Cette signature est contenue dans un fichier .cat. Mais un même fichier .cat peut contenir les signatures de plusieurs pilotes.

Certains pilotes comme HTTP.SYS n'ont pas de fichier .inf (bien que fournis par MS).

Un fichier inf peut faire référence à un autre fichier inf (chargement d'un autre pilote, inclusion de commandes ...)

Principe d'Installation d'un pilote avec fichier « .inf »

L'installation d'un pilote se déroule en deux temps :

- Pré-chargement :

Contexte : Winpe est inactif, le fichier boot.wim est monté dans un répertoire de travail

Le Pré-chargement du pilote consiste à copier les fichiers dans le « magasin » de WinPe (répertoire « DriversStore » et « CatRoot ») et à modifier diverses ruches dont « Drivers »

La signature est vérifiée dès le pré-chargement.

Deux méthodes :

	On isole les fichiers nécessaires, inf, sys, etc.
--	---

Avec DISM	<p>On identifie les fichiers « .cat » nécessaires avec « signtool.exe » On monte le fichier boot.wim dans un répertoire DISM.exe /Mount-Wim /WimFile:%ImageWimpe% /index:1 /MountDir:%Mount% On installe le pilote depuis la source contenant les fichiers utiles Dism /image:%Mount% /Add-Driver /Driver:C:\winpe10\hdaudio.inf Dism vérifie la signature des fichiers Dism met à jour le répertoire « DriverStore » et la ruche « Drivers » On copie les fichiers « .cat » nécessaires Si besoin, on recopie les fichiers associés et on modifie les ruches</p>
WinBuilder	<p>Il reprend « manuellement » l'activité de Dism.exe. Je suppose qu'elle consiste à isoler les éléments utiles dans les ruches « Drivers », « System », etc, et dans les répertoires comme « driversStore »...</p>

- Installation :

Contexte : Winpe actif

Après le démarrage de Winpe, le chargement peut être :

- automatique si le pilote gère un périphérique PNP
- ou manuel comme pour hdaudio.sys. Dans ce cas, on peut utiliser « DrvLoad.exe ».

note : Le chargement de hdaudio.sys installe le pilote hdaudio.sys automatiquement.

L'installation est réalisée par l'OS WinPe. Les fichiers sont recopiés depuis le magasin vers le répertoire des pilotes (...system32\drivers\).

Principe d'installation d'un pilote ne possédant pas de fichier .cat

Il arrive que certains pilotes « inbox » ne possèdent pas de référence à un fichier .cat dans le fichier .inf. C'est le cas pour les pilotes d'imprimantes PDF et XPS par exemple. La seule méthode que j'ai trouvé, c'est d'utiliser DISM. Il faut alors que l'OS actif soit windows 10. Dism réalise la pré-installation (import dans le driverstore). L'installation dans Winpe actif est à traiter au cas par cas.

Principe d'installation d'un pilote ne possédant pas de fichier .inf

Il faut construire les clés dans « ...system\controlset001\services\... ». Et il ne faut pas oublier le fichier .cat. Voir le script et l'installation de Http.sys, NativeWifiP, Vwiflft.

Comment trouver le fichier .cat des pilotes ?

Lu dans un script de Winbuilder de Win10PeSe :

```
// The cat file can be found by using the signtool.exe from the Windows SDK 8.0, use :
« signtool verify /kp /v /a c:\windows\system32\drivers\monitor.sys »
```

MS fournit le programme **signtool.exe** dans les SDK. Mais je ne sais pas comment il est arrivé sur mon PC. Avec un Visual Studio peut être...

Par exemple, pour trouver le fichier .cat de hdaudio.sys dans l'OS actif :

```
"c:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Bin\signtool.exe" verify /kp /v /a
c:\windows\system32\DriverStore\FileRepository\hdaudio.inf_amd64_dab2294dc8af0030\HdAudio.sys

Verifying:
c:\windows\system32\DriverStore\FileRepository\hdaudio.inf_amd64_dab2294dc8af0030\HdAudio.sys

File is signed in catalog: C:\WINDOWS\system32\CatRoot\{F750E6C3-38EE-11D1-85E5-
00C04FC295EE}\Microsoft-Windows-Client-Drivers-drivers-
Package~31bf3856ad364e35~amd64~~10.0.10240.16384.cat

Hash of file (sha1): 4417D4DE1E79592F6B38315112935CC87DE46C53
```

On peut aussi trouver le fichier .cat d'un pilote dans la référence (fichier iso de windows 10) montée dans un répertoire (clic droit "monter"):

☛ Attention : Il faut absolument que le build de l'OS actif dans lequel on lance signtool.exe soit le même que celui de la référence utilisée.

Exemple :

```
"c:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Bin\signtool.exe" verify /kp /v /a
C:\z_Win_OS\w10\mount\windows\system32\drivers\vwifibus.sys
```

Consulter les logs

La consultation des logs est primordiale lors des investigations. Il faut donc localiser les log utiles.

On lance souvent DISM pour construire WinPe. Et on lance DvrLoad après le démarrage de WinPe.

Pour Dism.exe	Pour DrvLoad.exe
C:\windows\inf\setup.dev.offline.logDism	X:\windows\inf\setupapi.dev.log
Dism affiche le chemin de sa 'log'	

Lister les pilotes : driverquery

driverquery /?

Affichage des pilotes signés uniquement : driverquery /si

Détails pour les autres Pilotes : driverquery /v /fo csv

Et un meilleur affichage avec PS :

```
driverquery /v /fo csv > c:\temp\qq.txt
```

```
Import-Csv c:\temp\qq.txt | Out-GridView
```

Tracer le chargement des pilotes lors du démarrage de Winpe

Modifier le Bcd ainsi :

```
Bcdedit /store ...\boot.bcd /set {default} bootlog Yes
```

Le fichier x:\windows\NtBtLog.txt contient la liste des pilotes chargés et non chargés. Mais sans explication. C'est parfois un début de piste.

Installation des pilotes hdaudio

L'installation des pilotes audio d'un constructeur augmente la taille de « Boot.wim » et donc le temps de chargement de Winpe. Ces pilotes spécifiques ne semblent pas toujours nécessaires.

Il semble que le pilote « hdaudio.sys » suffise dans bien des cas pour obtenir les sons dans Winpe. Son installation nécessite quelques commentaires.

- Le fichier hdaudio.inf ne possède pas de référence à un fichier .cat, donc « drvload » ne peut pas le charger sous Winpe
- La simple copie des .cat dans le répertoire « catroot » ne suffit pas pour son installation : il faut aussi que la ruche « drivers » contienne une référence à ce pilote.

Un test rapide : prendre la ruche « driver » de install.wim (mais elle ne contient pas le pilote « fbfw » pour écrire dans X :)

- Peut on le pré-installer avec Dism ?

Oui avec winpe inactif : `Dism /image:%Mount% /Add-Driver /Driver:C:\winpe10\hdaudio.inf`

Non avec winpe actif : le mode «online » n'est pas supporté par Dism pour Winpe !

- Peut on ensuite le charger avec DrvLoad avec winpe actif ? Oui
- Faut il quelques commandes supplémentaires après le démarrage de winpe? Oui.
- Les codecs sont visibles avec Msinfo.exe
- Le pilote MMCSS.SYS est il obligatoire ? Pas dans mon contexte !

Tous les formats « audio » ne sont pas reconnus. Par exemple, un fichier au format MP4 n'est pas lu.

Commandes supplémentaires post-démarrage

J'ai fait le choix de pré-charger le pilote hdaudio.sys. Avec une autre méthode d'installation (proche de WinPeSe) la pose desACL ne serait pas nécessaire.

La séquence suivante est nécessaire après le démarrage de Winpe.

- démarrage du service AudioEndpointBuilder
- chargement du pilote hdaudio :`drvload d:\sourceDesAjouts\audio\hdaudio\hdaudio.inf`
- modification des acl de la clé :

'HKLM:\SOFTWARE\MICROSOFT\Windows\CurrentVersion\MMDevices\Audio\Render'

Cette clé est créée après le démarrage du service AudioEndpointBuilder.

- démarrage du service audiosrv
- Enregistrement COM pour wmpplayer

WMPLAYER : build 10586=OK, build 14393=NOK pour Wav

J'ai testé les fichiers « .WAV » et « .MP3 ».

N'ayant pas pris le temps de modifier certaines clés, j'ai recopié le répertoire 64bits de Wmplayer de la source dans le répertoire « program files (X86) ». En effet, c'est dans ce répertoire que l'OS va chercher le programme de WmPlayer lors d'un double clic sur un fichier « audio ».

Bizarre mais je ne prends pas le temps de chercher une meilleure solution.

Note : avec le build 14393, il est impossible de lire les fichiers .Wav. Mais on peut lire les fichiers MP3.

API mciSendString de WinMm

Il existe diverses manières d'écouter de la musique. Cette API est facilement mise en œuvre dans un script PS avec un peu de C#.

J'ai rajouté dans boot.wim quelques fichiers et une clé dans « drivers32 » (voir le script si besoin).

Voir ces liens :

[https://msdn.microsoft.com/en-us/library/windows/desktop/dd757161\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd757161(v=vs.85).aspx)

pour les numéros des erreur: [https://msdn.microsoft.com/en-us/library/aa228215\(v=vs.60\).aspx](https://msdn.microsoft.com/en-us/library/aa228215(v=vs.60).aspx)

pour la signature c# : <http://pinvoke.net/default.aspx/winmm.mciSendString>

Différents sites signalent une anomalie avec le commande « open » et le type « mpegvideo » utilisé pour MP3. Mais après quelques essais, le script PS permet de lire des fichiers Wav et MP3. Et cela même si le nom du fichier contient des espaces.

Exemple en PS :

```
#
# permet de lire des MP3 et aussi des WAV dans WINPE
#
param([string] $fileAudio="")

Add-Type -Name mci -Namespace media -MemberDefinition @"
    [DllImport("winmm.dll", EntryPoint = "mciSendStringW", CharSet = CharSet.Unicode,
SetLastError = true, ExactSpelling = true)]
    public static extern int mciSendString(string lpstrCommand, string lpstrReturnString, Int32
uReturnLength, IntPtr hwndCallback);
"@
```

```

if ( $fileAudio -eq "" -or -not(test-path $fileAudio) ){
    "fichier absent : $fileAudio"
    return
}

switch (([System.io.path]::getextension($fileAudio)).toupper())
{
    ".WAV"      {
                $type="waveaudio"
                break
            }
    ".MP3"      {
                $type="mpegvideo"
                break
            }
    default {
                "extension non traitée"
                return
            }
}

# ok with spaces in the name of file
$stringOpen = "open " + ""$fileAudio`"" + " Type $type Alias playsound" ;
$stringOpen=$stringOpen
$mediaError = [media.mci]::mciSendString($stringOpen, "", 0, 0);
"mediaError=$mediaError"
$stringPlay = "play playsound wait"
$stringPlay=$stringPlay
$mediaError = [media.mci]::mciSendString($stringPlay, "", 0, 0)
"mediaError=$mediaError"
$stringClose = "close playsound wait"
$stringPlay=$stringClose
$mediaError = [media.mci]::mciSendString($stringClose, "", 0, 0)
"mediaError=$mediaError"

```

Wifi

Mes références

<http://ploadletter.co.uk/2011/12/03/windows-pe-builder-script-for-waik-including-wifi-support/>

<http://www.serverwatch.com/server-tutorials/using-netsh-commands-for-wi-fi-management-in-windows-8.html>

Pour décrypter :

<http://stackoverflow.com/questions/10765860/decrypt-wep-wlan-profile-key-using-cryptunprotectdata>

<http://superuser.com/questions/133097/netsh-wlan-add-profile-not-importing-encrypted-passphrase>

Autres sites:

http://blogs.technet.com/cfs-filessystemfile.ashx/_key/telligent-evolution-components-attachments/01-

[6127-00-00-03-31-62-58/Windows-7-Deployment-Procedures-in-802-1X-Wired-Networks.pdf](http://www.msfm.org/board/topic/162453-winpe-40-enable-wireless-support/)

<http://www.msfm.org/board/topic/162453-winpe-40-enable-wireless-support/>

Que faut il installer ?

Pour Wifi, il y a plusieurs composants à installer :

- les pilotes des cartes fournis par les OEM
j'ai choisi de faire un pré-chargement avec DISM
- les pilotes utilisés dans la couche "réseau"
 - NativeWifiP
 - Vwifflt
 - VwifiBus : il est installé automatiquement par l'include netvwifibus.inf
- les services de la couche réseau
 - MS_NativeWifiP
 - MS_vwifi

J'ai choisi de faire un pré-chargement avec DISM. Et ils seront installés avec netcfg.exe

Identifier et collecter les fichiers inf, sys, cat ...

- Les NetServices pour Netcfg.exe
 - Netwifi.inf pour le service MS_NativeWifiP
 - Netvwifflt.inf pour le service MS_Vwifi
- Le pilote de ma carte wifi (intel dans mon cas)
 - NetWew00.inf, NetWew00.sys, NetWfw00.dat

Le fichier .inf du pilote de ma carte possède un 'include' :

voir le log «...\windows\inf\setupapi.offline.log »

- Pour l'include:
 - NetWifiBus.inf, VwifiBus.sys, VwifiBus.sys.mui (fr-FR ou En-us si beta)
 - * La commande "dism /add-driver" ira chercher le pilote dans le répertoire \inf :
%mount%\windows\inf\vwifibus.sys
- Donc copier au préalable le fichier NetWifiBus.inf dans ce répertoire.

Copier le répertoire L2Schemas

Ce répertoire est disponible dans la référence de l'ISO.

Les commandes NetSh élémentaires

Il est parfois utile d'exporter ses profils Wifi depuis son windows10 et de les importer dans son Winpe. Dans une machine Win10, configurer les connexions Wifi. Puis, exporter ces configurations dans des fichiers Xml. Ils contiendront les SSID et les mots de passe.

Après le démarrage de WinPe, il faudra importer le profil souhaité. Puis se connecter.

- Export des profils enregistrés avec mot de passe en clair dans le répertoire courant :

```
netsh wlan export profile key=clear
```

- Export d'un profil :

```
netsh wlan export profile name="freebox_opfm" folder="%SourceDriversWifi%\Profils" key=clear
```

- Import d'un profil dans Winpe :

```
netsh wlan add profile filename="x:\MesProfilesWifi\freebox_opfm"
```

- Connexion wifi avec le profil importé dans Winpe :

```
netsh wlan connect name="freebox_opfm" ssid="freebox_opfm"
```

Un script de connexion Wifi en PS

Sur le site CodePlex <http://managedwifi.codeplex.com/> j'ai trouvé un code C# qui permet la connexion Wifi. Je l'ai placé dans un script PS. J'ai ajouté une interface graphique ainsi que la fonction de déconnexion.

Sur mes PC, je constate qu'il faut désactiver/activer la carte Wifi avant la connexion. Sans cela, l'identification du réseau échoue. Je n'ai pas d'explication.

Les imprimantes dans Winpe

Les imprimantes sont très différentes les unes des autres (multi-fonctions ou non, laser ou jet d'encre, etc). Elles peuvent être raccordées de diverses manières, réseau, usb, wifi, web... L'architecture des pilotes a évolué au fil des versions de l'OS (mode kernel, mode user ...).

Trouver le bon pilote est un point délicat. Il est plus simple de laisser le système trouver tout seul les pilotes. C'est ce qui est fait avec les imprimantes partagées sur un serveur. Lors de l'installation de l'imprimante, les pilotes contenus dans le serveur sont automatiquement téléchargés dans la station de travail.

Dans le cas de l'installation d'une imprimante USB, il faudra mettre à disposition de Winpe les pilotes nécessaires soit en les embarquant dans le fichier « boot.wim » soit dans un autre support, USB par exemple.

Je ne dispose que d'une seule imprimante, ce qui limite mes investigations. Je peux la partager sur le réseau ou la raccorder en USB.

➤ Site de référence de l'architecture : <https://technet.microsoft.com/en-us/library/cc976755.aspx>

➤ pour lancer l'ihm d'ajout d'imprimante, taper :

« "X:\windows\System32\rundll32.exe" printui.dll,PrintUIEntryDPIAware /il »

➤ consulter le fichier de log : « x:\windows\system32\inf\setupapi.dev.log »

Imprimantes invisibles

☛ Les imprimantes sont bien visibles dans notepad (menu « imprimer ») mais elles n'apparaissent pas dans le panneau de configuration

Elles sont visibles avec powershell, Get-printer et gmi Win32_printer.

Particularités de mon imprimante

Il s'agit de l'imprimante « samsung SCX-4500 Series ». Les pilotes de la version 64 bits utilisent néanmoins des programmes en 32 bits. Donc WOW64 est obligatoire dans ce cas. Pour l'instant, je ne m'intéresse pas à la fonction « scanner ».

Imprimante réseau dans la session « administrateur » : ca marche

Avec DISM, j'ai installé les « composants InBox» NTPRINT.INF et USBPRINT.INF.

Après avoir copié des fichiers et des clés de registre, le service « spooler » démarre.

A partir du « panneau de configuration/devices and printers », on peut lancer l'installation de l'imprimante réseau. Il faut fournir les « credential » d'un utilisateur autorisé à gérer l'imprimante et la file d'impression. Bien vérifier les divers paramètres du partage de l'imprimante sur le serveur.

Imprimante réseau dans la session « system » : impossible

L'installation débute normalement. Le gros fichier .cab est bien téléchargé. Mais la finalisation (texte « finalisation » apparaissant dans la fenêtre d'installation) déclenche l'affichage « connexion impossible à l'imprimante : erreur 0x00000534 ».

L'analyse de la trace procmon et la comparaison avec une installation sur un autre PC confirment bien le téléchargement des pilotes et la divergence lorsque LSA consulte la SAM. Le programme « PrintIsolationHost.exe n'est pas lancé dans le cas de l'erreur.

L'activité de « Spoolsv.exe » à ce moment là :

- saisie du nom du compte et du mot de passe de l'utilisateur créé sur le serveur
- création de la clé de registre « \software\microsoft\windowsnt\CurrentVersion\print\providers\client Side Rendering Print Providers\servers\...\Client Side Port\... ».
- téléchargement du fichier .cab contenant les pilotes
- affichage « finalisation ...» (pas toujours visible)

- suppression de la clé précédemment créée
- affichage du message d'erreur « Erreur 0x00000534 »

Lors d'une recherche sur le site MSDN des codes d'erreur, j'ai trouvé ceci :

« error 0x00000534 : No mapping between account name and SID was done ».

Pourquoi ce contrôle ? Peut être parce qu'une imprimante est un attribut d'un compte (un peu comme dans l'AD). Test avec « SystemSetupInProgress » ???

Impossible pour moi d'aller plus loin.

Site Msdn décrivant les erreurs : <https://msdn.microsoft.com/en-us/library/cc231199.aspx>

Imprimante USB dans la session « administrateur »

En installant cette imprimante USB sur un autre pc, et en analysant le fichier « ...\\inf\\setupdev.log », on trouve le répertoire contenant les bons pilotes dans le « driverstore ».

Cela permet ensuite de tenter l'installation sur Winpe, soit avec le panneau de configuration, soit à partir du gestionnaire de périphériques en choisissant l'imprimante et non le scanner dans mon cas.

L'installation réussit. L'imprimante est visible dans « notepad/menu imprimer ».

Mais l'impression échoue avec le message « Un appel à StartDocPrinter n'a pas été opéré ».

La trace procon montre que le spooler (spoolsv.exe) teste la clé « SystemSetupInProgress ».

Le contournement :

- SystemSetupInProgress = 0
- démarrage du service spooler et lancement de l'impression au moins une fois, avant StartDocPrinter
- SystemSetupInProgress = 1

L'impression est opérationnelle. Mais là aussi, l'imprimante n'est pas visible dans le « panneau de configuration/devices and printers ».

Il est plus simple d'installer l'imprimante depuis le gestionnaire de périphériques après l'avoir connecté (évite la création d'un port et d'une imprimante fantôme).

L'imprimante USB installée dans une session est visible et disponible dans l'autre session.

Imprimante USB dans la session « system »

L'installation et l'impression fonctionnent comme dans le cas de la session « administrateur ».

Le scanner de ma « samsung » : Traitement manuel obligatoire

A la fin de l'installation de l'imprimante et du scanner, certaines clés ne sont pas renseignées. Il faut donc, dans l'état actuel, ajouter les clés absentes et arrêter/redémarrer le service « stisvc ».

Les clés à ajouter :

HKLM\SYSTEM\CurrentControlSet\Control\Class\{6bdd1fc6-810f-11d0-bec7-08002be2092f}\0000

```
"SubClass"="StillImage"
"friendlyname"="My scan Samsung"
"DeviceID"="{6bdd1fc6-810f-11d0-bec7-08002be2092f}\\0000"
"capabilities"=dword:00000031
"deviceType"=dword:00000001
"deviceSubType"=dword:00000001
```

☛ La présence d'un index (ici **0000**) complique l'automatisation.

☛ L'architecture de WIA :

<https://msdn.microsoft.com/en-us/windows/hardware/drivers/image/wia-architecture-overview>

<https://msdn.microsoft.com/en-us/windows/hardware/drivers/image/wia-core-components>

☛ L'activation des traces de WIA fournit des informations utiles pour la mise au point

Le fichier de log : \debug\wiadebug.log.

La modification des clés :

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\StillImage\Trace\wiaservc.dll]
```

```
"TraceFlags"=dword:00000007
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\StillImage\Trace\sti.dll]
```

```
"TraceFlags"=dword:00000007
```

Les imprimantes PDF et XPS

C'est le spooler qui installe entièrement et automatiquement ces deux imprimantes. Après le démarrage du service spooler, il faut attendre 2 minutes avant que ne débute automatiquement le lancement de l'installation.

Dans le gestionnaire de périphériques, les imprimantes PDF et XPS se situent dans l'item « imprimantes » et avec le nom « périphérique inconnu ». Pour obtenir un affichage plus correct, on peut lancer la commande :

```
DRVLOAD X:\windows\System32\DriverStore\FileRepository\printqueue.inf_amd64_293dcb0d10d72f40\printqueue.inf
```

Ensuite, les imprimantes sont déplacées dans l'item « print queue » et retrouve leur nom correct.

Note : activité du spooler constatée avec procmon lors de l'installation automatique de ces deux imprimantes

```
installation des 2 packages
    HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\PackagesToAdd\
        ntprint et prnms003
lancement de
    X:\windows\system32\regsvr32.exe /s "X:\windows\system32\spool\drivers\x64\3\PrintConfig.dll"
installation du service PrintNotify
    HKLM\SYSTEM\CurrentControlSet\Services\PrintNotify
    X:\windows\system32\spool\drivers\x64\3\PrintConfig.dll
```

recherche des packages :

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\PackagesToAdd\

...

le service "DeviceInstall" (umpnpmgr.dll) lance Drvinst.exe

HKLM\SYSTEM\CurrentControlSet\Services\DeviceInstall\Parameters

%SystemRoot%\system32\umpnpmgr.dll

« **DrvInst.exe** "4" "d" "X:\windows\system32\spool\tools\Microsoft Print To PDF\prnms009.Inf" "9" "4cf18dca3" "0000000000000670" "WinSta0\Default" "00000000000003AC" "208" "X:\windows\system32\spool\tools\Microsoft Print To PDF" »

qui appelle le service Windows Update absent : X:\windows\System32\wuaueng.dll

qui écrit la cle HKLM\DRIVERS\DriverDatabase\DeviceIds\{4d36e979-e325-11ce-bfc1-08002be10318}\

activité du service HKLM\SYSTEM\CurrentControlSet\Services\CryptSvc

copie du fichier cat

src :X:\windows\system32\DriverStore\FileRepository\prnms009.inf_amd64_bd3f6a64dee1535d\prnms009.cat

dst : X:\WINDOWS\SYSTEM32\CATROOT\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\oem14.cat

son contrôle : X:\windows\system32\catroot2\dberr.txt

En même temp, construction du gros fichier

X:\windows\system32\spool\drivers\x64\PCC\ntprint.inf_amd64_7b3eed059f4c3e41.cab

Ecriture de la clé décrivant les caractéristiques de l'imprimante pdf

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\Printers\Microsoft Print to PDF

Construction du fichier cab

X:\windows\system32\spool\drivers\x64\PCC\prnms009.inf_amd64_bd3f6a64dee1535d.cab

Création des fichiers

X:\windows\system32\spool\V4Dirs\50C45F04-5F15-4B72-A323-05EBFFE9D913\device_bidi.gpd

X:\windows\system32\spool\V4Dirs\50C45F04-5F15-4B72-A323-05EBFFE9D913\merged.gpd

...

lecture de HKLM\System\CurrentControlSet\Control\Print\EnableJavaScriptDebugging

printconfig.dll appelle jscrip.dll

activité impossible à identifier

création de HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\OfflinePrinters\Fax

suppression de HKLM\SOFTWARE\Microsoft\Windows

NT\CurrentVersion\Print\OfflinePrinters\Microsoft Print To PDF

lancement de « **DrvInst.exe** "1" "0" "SWD\PRINTENUM\{024AFB8F-1EF1-4ECF-B5EF-27F41CC0E988}" "" "" "49d499bbb" "0000000000000000" »

lecture de HKLM\DRIVERS\DriverDatabase\DriverPackages\printqueue.inf_amd64_293dcb0d10d72f40

lancement de « **DrvInst.exe** "4" "d" "X:\windows\system32\spool\tools\Microsoft XPS Document Writer\prnms001.Inf" "9" "4ec392ceb" "000000000000067C" "WinSta0\Default" "0000000000000678" »

```
"208" "X:\windows\system32\spool\tools\Microsoft XPS Document Writer" »
...
recherche de X:\WINDOWS\SYSTEM32\FaxPrinterInstaller.dll

lancement de « DrvInst.exe "1" "0" "SWD\PRINTENUM\{88C0C652-C649-4C6D-8136-5F8B16982552}" "" "" "455aafd0f" "0000000000000000" »

lecture de HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Device Installer\DebugInstall
```

Winpe « mode Flat » dans un VHD pour une VM hyperV

Veillez consulter les sites suivants pour obtenir des informations sur :

- le « mode flat » de Winpe : <https://technet.microsoft.com/en-us/library/hh825045.aspx>
- les vhd : [https://msdn.microsoft.com/en-us/library/windows/desktop/dd323654\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd323654(v=vs.85).aspx)
- VHD et native boot : <https://msdn.microsoft.com/fr-fr/windows/hardware/commercialize/manufacture/desktop/boot-to-vhd--native-boot--add-a-virtual-hard-disk-to-the-boot-menu>

Pour utiliser Winpe dans un VHD, vous devez construire un BCD adéquat et ajouter le paramètre « Winpe Yes ». C'est simple et pourtant

Ce site traite de la commande « bootsect.exe /nt60 » :

<https://blogs.technet.microsoft.com/heyscriptingguy/2015/12/04/build-a-powershell-enabled-windows-pe-key-part-5/>

Contexte utilisé pour les commandes de cet exemple

Les chemins sont les suivants :

Boot.Wim	C:\MicroWinpeBuilder\media\sources\boot.Wim
Fichier VHD	C:\Vhd\Winpe.vhd note : créer le répertoire si besoin
Unité pour le VHD	« H : », pour monter le VHD (clic droit ou commande PS, ou Dispart ou le gestionnaire de disque...)
BCD à modifier	H:\boot\bcd
Bios	Partition Bios en non UEFI

Les cinq étapes de la construction



1 - construire le disque VHD (ou vhdx) et le monter : plusieurs outils sont possibles

- avec Diskpart et les commandes suivantes :

```
create vdisk file="C:\Vhd\Winpe.vhd" maximum=9000 type=fixed
attach vdisk
create partition primary
assign letter=H
format fs=ntfs quick
active
exit
```

Pour utiliser un fichier de script avec diskpart :

```
copier ce texte dans un fichier « CreateVhd.txt »
lancer la commande « diskpart /s CreateVhd.txt »
```

- avec des commandes PS :

voir <https://technet.microsoft.com/en-us/library/hh848503.aspx>

```
$vhdPath = "C:\Vhd\Winpe.vhd"
$vhdSize = 9GB
$vhdLetter="H"
New-VHD -Path $vhdPath -Fixed -SizeBytes $vhdSize | Mount-VHD -Passthru | Initialize-Disk
-PartitionStyle MBR -Passthru | New-Partition -DriveLetter $vhdLetter -UseMaximumSize -IsActive
| Format-Volume -FileSystem NTFS -Confirm:$false -Force
```



2 - décompresser Boot.wim dans le VHD : au moins 2 outils efficaces

- avec 7z.exe version supérieure ou égale à 9.20

Utiliser l'interface graphique pour décompresser le fichier Boot.wim dans H:\

- Avec Dism

```
dism /Apply-Image /ImageFile:"C:\MicroWinpeBuiler\media\sources\boot.wim" /Index:1 /ApplyDir:H:\
```



3 - construire les fichiers de démarrage (legacy et UEFI) avec Bootbcd.exe

```
BCDboot H:\Windows /s H: /f ALL
```



4 - modifier le BCD avec bcdedit.exe

Le BCD a été créé par la commande BcdBoot. Mais cette commande ignore Winpe. Donc on ajoute ce paramètre.

```
Bcdedit /store H:boot\bcd /set {default} Winpe YES
et avec Powershell :
Bcdedit /store H:boot\bcd /set `{default}` Winpe YES
```

On obtient le BCD suivant :

```
PS C:\WINDOWS\system32> Bcdedit /store H:boot\bcd
```

Gestionnaire de démarrage Windows

```
-----
identificateur    {bootmgr}
device           partition=H:
description      Windows Boot Manager
locale          en-us
inherit          {globalsettings}
default          {default}
displayorder     {default}
toolsdisplayorder {memdiag}
timeout         30
```

Chargeur de démarrage Windows

```
-----
identificateur    {default}
device           partition=H:
path             \Windows\system32\winload.exe
description      Windows PreInstallation Environment
locale          en-us
inherit          {bootloadersettings}
allowedinmemorysettings 0x15000075
osdevice        partition=H:
systemroot      \Windows
nx              OptOut
winpe          Yes
```

5 – éjecter le VHD

- avec explorer : clic droit puis « éjecter »
- avec Diskpart :

```
diskpart
select vdisk file="C:\Vhd\Winpe.vhd"
detach vdisk
exit
```

- avec PS :

```
Dismount-VHD -Path "C:\Vhd\Winpe.vhd"
```

Winpe « mode Flat » dans un VHD pour disque dur

Comme précédemment. Seul le BCD est différent. De plus, il n'est pas situé au même endroit.

En effet, puisque l'on va démarrer le PC depuis un disque dur (externe ou non, usb ou non), le BCD est situé dans le disque dur et non dans le VHD. Le VHD est un simple fichier de grande taille contenu dans ce disque dur. De plus, le disque dur peut avoir plusieurs partitions ou non.

Exemple de mon disque dur USB contenant trois partitions F, I et J., ainsi que plusieurs Winpe.

Disque 1 De base 465,76 Go En ligne				
	popReserved	winpe (I:) ←	Maison (J:) ←	
	1000 Mo NTFS Sain (Actif, Pa	9,77 Go NTFS Sain (Partition princ	415,95 Go NTFS Sain (Partition principale)	39,06 Go Non alloué

Le BCD présent dans mon disque dur USB :

```
C:\WINDOWS\system32>bcdedit /store f:\boot\bcd
```

Gestionnaire de démarrage Windows

```
-----
identificateur      {bootmgr}
description         Windows Boot Manager
locale              fr-fr
inherit              {globalsettings}
default              {default}
displayorder        {default}
                    {734b7449-e4bb-11e1-a114-001e330ca2a8}
                    {00a5bd49-8f29-11e6-9663-c01885b223e6}
toolsdisplayorder  {memdiag}
timeout             30
```

Chargeur de démarrage Windows

```
-----
identificateur      {default}
device            vhd=[I:]\vhd-winpe4-NB.vhd
path                \windows\system32\boot\winload.exe
description         vhd-winpe4-NB
locale              fr-fr
inherit              {bootloadersettings}
osdevice         vhd=[I:]\vhd-winpe4-NB.vhd
systemroot          \windows
detecthal           Yes
winpe             Yes
ems                 No
```

Chargeur de démarrage Windows

```
-----
identificateur      {734b7449-e4bb-11e1-a114-001e330ca2a8}
device            ramdisk=[I:]\sources\boot.wim,{7619dcc8-fafe-11d9-b411-000476eba25f}
```

```

path                \windows\system32\boot\winload.exe
description         winpe10 build 14393 sur partition2
locale              fr-fr
inherit             {bootloadersettings}
osdevice            ramdisk=[I:]\sources\boot.wim,{7619dcc8-fafe-11d9-b411-000476eba25f}
systemroot          \windows
detecthal           Yes
winpe               Yes
ems                 No

```

Chargeur de démarrage Windows

```

-----
identificateur      {00a5bd49-8f29-11e6-9663-c01885b223e6}
device             vhd=[J:]\mesvhd\vhd2016\test.vhd
path                \windows\system32\boot\winload.exe
description         MicroWinpeBuilder 14393
locale              fr-fr
inherit             {bootloadersettings}
osdevice            vhd=[J:]\mesvhd\vhd2016\test.vhd
systemroot          \windows
detecthal           Yes
winpe               Yes
debug               Yes
ems                 No

```



Attention à propos de ramdisk:

Ce n'est parce que vous écrirez « **ramdisk=[I:]\sources\boot.wim,{7619dcc8-fafe-11d9-b411-000476eba25f}** » que le fichier « ...\boot\bcd » contient les informations relatives à l'entrée de registre « **{7619dcc8-fafe-11d9-b411-000476eba25f}** ». Assurez vous soit de la provenance du fichier « BCD » soit de la présence de cette entrée.

Ce site de MS explique ce point ainsi que la manière de vérifier cette entrée.

<https://social.technet.microsoft.com/Forums/en-US/f2f2cabe-27b4-4cea-bf1d-76d4cea7ccbf/alphabetic-object-in-the-dvd-bcd?forum=win10itprosetup>

```

bcdedit /store "C:\Users\Balubeto\Downloads\DVD\boot\bcd" /v /enum all

```

```

...

```

```

Device options

```

```

-----
identifieur         {7619dcc8-fafe-11d9-b411-000476eba25f}
ramdiskdevice       boot
ramdiskpath         \boot\boot.sdi

```


Cette entrée n'est pas présente dans le BCD situé sur le disque dur de votre PC.

Des compléments sur winpe

« install.ESD » : 7Z.exe version 64bits supérieure à 15

Je n'ai pas d'information sur ce format nouvellement utilisé par MS.

La version bêta de 7Z.exe 64bits permet d'ouvrir les fichiers compressés « install.ESD ».

Le boot pxe de winpe avec tftp32 version 64bits de « Jounin »

C'est plutôt lent si l'image boot.wim a une taille de 500Mo ou plus.

Télécharger le serveur TFTP de « Jounin ».

<http://tftpd32.jounin.net/download/tftpd64.452.zip>

On pourrait aussi utiliser le DHCP d'un serveur MS.

Il faut récupérer des fichiers dans l'image boot.wim. Donc monter cette image. Et copier les fichiers de <mount>\windows\system32\boot\pxe dans un répertoire qui sera la racine du serveur TFTP. Puis copier le contenu de la clé usb, donc toute l'arborescence des fichiers de votre winpe final (boot, sources,etc) dans cette racine du serveur TFTP.

La configuration du serveur TFTP est assez simple : cocher serveur TFTP et serveur DHCP, renseigner le fichier à charger « pxeboot.n12 », mettre un ip fixe éventuellement sur le pc hébergeant le serveur tftp, renseigner le répertoire de base (celui qui héberge tous les fichiers précédents), ouvrir l'onglet de log.

Penser à ouvrir « domaine public » dans le firewall pour TFTP, ce qui devrait être fait car le premier lancement de TFTP64 de « jounin » ouvre ces ports (UAC?), et le firewall aurait dû vous avertir (selon votre configuration bien sûr).

En gros : connecter les deux pc directement par un câble « rj45 » (pas besoin de câble croisé avec la norme gigabyte de tous les pc modernes), modifier le « bios » pour activer le « boot pxe ». Lors du démarrage du pc, on choisit « boot pxe ». Après avoir demandé un adresse ip au serveur DHCP, la carte envoie une trame TFTP sur le réseau. Le serveur TFTP sur l'autre pc la reçoit et lui renvoie le fichier « pxeboot.n12 » (n12 ou com selon que l'on veut agir ou non sur F12 en cours de route mais je ne me souviens à quoi cela sert). Puis le pc exécute ce programme. Ce dernier charge alors « bootmgr.exe » depuis le pc serveur TFTP. Il s'agit bien de « bootmgr.exe » et non de « bootmrg » de la racine de la clé usb ! Puis « bootmgr.exe » charge à son tour le BCD, puis tout le reste (boot.sdi, ... \sources\boot.wim) depuis le serveur TFTP. C'est long mais ça marche.

Le RamDisk FBWF.SYS : ScratchSpace limité à 512Mo ?

Pour modifier cette valeur, on peut utiliser DISM. Mais si on tente :

```
« dism /image:%Mount% /Set-ScratchSpace:1024
```

```
Valeur d'espace de travail non valide. Sélectionnez 32, 64, 128, 256 ou 512. »
```

Je n'ai pas trouvé d'autre moyen que de modifier le code du driver. Après la modification du code, il faut signer le fichier. Ne pas inverser les étapes !

Pour désassembler le code de fbwf

Pour désassembler le fichier du pilote, j'ai utilisé «PeBrows64 Pro». Consulter le site pour plus de détail :

<http://www.smidgeonsoft.prohosting.com/>

Il est gratuit (si j'ai bien compris l'anglais). Si je n'avais pas lu la documentation sur le site, je n'aurais jamais pu voir une ligne de code. Si j'y suis arrivé, tout le monde peut le faire.

Les symboles

Au premier lancement , le logiciel signale que la variable d'environnement « `_NT_SYMBOL_PATH` » est absente et qu'il faut utiliser le mode « en tant qu'administrateur ».

Quand on désassemble peu souvent, on perd vite les bons réflexes. J'avais oublié la nécessité des symboles pour rendre un listing compréhensible.

Consulter le site pour plus de détail :

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms681416\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms681416(v=vs.85).aspx)

Donc, dans une console 'CMD' je lance :

```
« set _NT_SYMBOL_PATH=svr*c:\localsymbols*http://msdl.microsoft.com/download/symbols »
```

Puis je lance

```
"C:\Program Files\SmidgeonSoft\PEBrowse64 Professional\PEBrowse64.exe"
```

En tâtonnant, et en alternant « clic droit/menu », on finit par trouver « `driverEntry` ». Puis on aperçoit l'appel à « `FbwfCheckForWinPEBoot` ». Cette fonction consulte la valeur « `WinPECacheThreshold` ». Elle réalise des tests de borne : `CMP EAX,0x400` où `0x400` correspond à 1000d et suivi par un 'JBE', on peut dire que c'est la limite à 512Mo.

Avec mes script PS, après avoir modifier le BCD de ma clé USB pour ajouter « TESTSIGNING » et avoir modifier le paramètre « `WinPECacheThreshold` » dans la ruche 'system', je fais la modification du programme avec un « `CMP EAX, 0xFF00` », l'injection du nouveau checksum et je signe le fichier, je le recopie dans boot.wim : échec !

Il y a peut être une modification de la valeur lue pour tenir compte de quelque chose comme la taille mémoire. Donc je relis le code et fouille un peu plus.

Dans la fonction, la valeur lue est stockée dans une variable en mémoire. Un clic droit de PeBrowser64 permet de trouver une option qui fournit les « cross reference », c'est à dire les divers endroit du fichier où cette case mémoire est utilisée. Je trouve rapidement la fonction « `FbwfInstanceSetup` » avec le texte « `FbwfInstanceSetup: Failed to adjust scratch space; status = 0x%X` ». Je fouille un peu. Un appel à « `ZwQuerySystemInformation` », un calcul, une nouvelle comparaison avec « `CMP RAX,0x40000000` » suivi d'un « JBE » vers le texte de l'erreur ...

Je modifie le « JBE » en JMP pour neutraliser le saut vers le code de l'erreur...

Et ça marche, la taille de X : est de 2Go dans le Winpe.

La signature pour test

Le plus difficile c'est de trouver les 3 programmes 'magiques' : Makecert.exe, CertMgr.exe, SignTool.exe

Je ne sais pas comment ils sont arrivés sur mon PC. Avec Visual Studio Community peut être.

Avec windows10 64bits, chaque pilote doit être signé soit avec un 'vrai' certificat soit avec un certificat de test.

Un développeur de pilote a 2 possibilités pour ses tests (répétitifs) :

- bloquer la vérification de la signature sur le PC de test

MS empêche l'installation d'un tel pilote non signé n'importe où et par n'importe qui. Le développeur doit absolument agir physiquement sur le PC de test pour bloquer la vérification du certificat d'un pilote avec :

- soit un appui sur F8 lors du démarrage et en sélectionnant l'option '?'
- soit en connectant un PC de débogage ... le chargement du pilote impose que le développeur appuie sur 'g' dans son PC

- utiliser un certificat de test déployé sur plusieurs PC de tests

Pour cela, il faut modifier le BCD avec l'option « TESTSIGNING ». Une information 'Mode test' apparaîtra en bas à droite sur l'écran. Ainsi l'utilisateur final est averti du danger.

Exemple : Bcdedit.exe /store ??:\boot\bcd -set {default} TESTSIGNING ON

Pour plus d'information sur « The TESTSIGNING Boot Configuration Option » :

<https://msdn.microsoft.com/en-us/library/windows/hardware/ff553484%28v=vs.85%29.aspx?f=255&MSPPError=-2147217396>

Note :

Before setting BCDEdit options you might need to disable or suspend BitLocker and Secure Boot on the computer.

Starting with Windows 7, Windows displays this watermark only in the lower left-hand corner of the desktop.

Pour signer le driver

Il faut créer un certificat de test, l'installer éventuellement sur la machine du développeur....

La création d'un certificat de test avec Makecert.exe

Pour plus d'information :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff548693\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff548693(v=vs.85).aspx)

MakeCert -r -pe -ss TestCertStoreName -n "CN=CertName" CertFileName.cer

Comment est il arrivé sur mon pc :

```
"C:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Bin\x64\MakeCert.exe" ?  
cd "C:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Bin\x64\  
MakeCert.exe -r -pe -ss NoelBlancStore -n "CN=CertificatTestingNoelBlanc" CertificatTestingNoelBlanc.cer
```

L'installation du certificat dans le pc du développeur

Pour plus d'information :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff553506\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff553506(v=vs.85).aspx)

Avant de signer un fichier, il faut installer le certificat sur la machine du développeur. Si le certificat a été généré sur la même machine, c'est inutile.

Il faut retenir que :

- pour signer un pilote : le certificat peut être installé dans n'importe quel store
- mais pour vérifier la signature d'un pilote signé avec ce certificat, ce certificat doit être installé dans : "Trusted Root Certification Authorities"

The name of the 'Trusted Root Certification Authorities certificate' store is root.

D'où mes commandes, en étant admin :

```
cd "C:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Bin\x64\  
CertMgr.exe /add CertificatTestingNoelBlanc.cer /s /r localMachine root
```

OK, je vois ce certificat avec certmgr.msc dans mon pc de 'dev'

Mais comment signer le pilote avec ce certificat ?

Test-Signing a Driver File :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff553467\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff553467(v=vs.85).aspx)

Test-Signing Driver Packages :

<https://msdn.microsoft.com/en-us/library/windows/hardware/ff553480%28v=vs.85%29.aspx?f=255&MSPPErr=-2147217396>

The following command shows how to use SignTool to test-sign a driver file.

This example embeds a signature in Toaster.sys, which is in the amd64 subdirectory under the directory in which the command is run.

The test certificate is named "contoso.com(test)" and it is installed in the certificate store named "PrivateCertStore."

```
SignTool sign /v /s PrivateCertStore /n contoso.com(test) /t
```

```
http://timestamp.verisign.com/scripts/timestamp.dll amd64\toaster.sys
```

D'où ma commande :

```
SignTool sign /v /s NoelBlancStore /n CertificatTestingNoelBlanc /t  
http://timestamp.verisign.com/scripts/timestamp.dll fbwf.sys
```

Installation du certificat sur la machine de test (non winpe)

Cette procédure est à utiliser dans le cas d'un windows non Winpe.

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff547618\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff547618(v=vs.85).aspx)

The test certificates that are used to embed signatures in driver files and to sign a driver package's catalog file must be added to :

- the Trusted Root Certification Authorities certificate store
- the Trusted Publishers certificate store.

Installing a Test Certificate on a Test Computer :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff553563\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff553563(v=vs.85).aspx)

The following CertMgr command adds the certificate in the certificate file CertificateFileName.cer to the Trusted Root Certification Authorities certificate store on the test computer:

```
CertMgr.exe /add CertificateFileName.cer /s /r localMachine root
```

The following CertMgr command adds the certificate in the certificate file CertificateFileName.cer to the Trusted Publishers certificate store on the test computer:

```
CertMgr.exe /add CertificateFileName.cer /s /r localMachine trustedpublisher
```

Installation du pilote modifié et nouvellement signé dans Winpe

C'est le cas qui m'intéresse.

C'est inutile d'installé le certificat.

Il faut néanmoins que le pilote soit signé.

Puis on modifie le BCD pour activer l'option "TESTSIGNING"

Et il ne faut pas oublié de modifier la valeur souhaitée pour la taille du RamDisk :

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\FBWF
```

```
WinPECacheThreshol=dword 00000400 si 1Go
```

```
WinPECacheThreshol=dword 00000800 si 2Go
```

Une collection de clés utiles et à compléter

HKEY_LOCAL_MACHINE\SYSTEM\setup

la valeur « SetupPhase » correspond aux phases décrites dans l'aide de AIK
la clé « AllowStart » : à documenter !

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\

la clé MiniNt indique que l'OS est un Winpe
la valeur SystemStartOptions contient les valeurs du BCD comme :
TESTSIGNING MININT ('winpe' dans le BCD) BOOTLOG et d'autres valeurs
inconnues.

Cette valeur est testée par le pilote 'Fbwf.sys' lors de son chargement qui cherche la chaîne
« MiniNT ». Ce pilote cherche aussi dans cette valeur les chaînes suivantes
'EnabledOnAllSkus' et 'WinpeCacheThreshold'.

La commande 'Dism.../set-ScratchSpace » modifie la valeur :
« \system\...\services\fbwf\WinpeCacheThreshold »

La clé « \system\...\services\PartMgr\parameters\ » contient la valeur « SanPolicy ». Elle permet de
masquer les disques locaux de la machine quand winpe est chargé. Elle est utilisée par « WinToGo ».

Des sites

Vhd : <http://go.microsoft.com/fwlink/?LinkId=205691>

Bcdedit :

<https://msdn.microsoft.com/en-us/windows/hardware/commercialize/manufacture/desktop/bcdedit-command-line-options>

[https://technet.microsoft.com/en-us/library/cc709667\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc709667(v=ws.10).aspx)

[https://msdn.microsoft.com/en-us/library/windows/hardware/dn653287\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/dn653287(v=vs.85).aspx)

Partie investigation au sein de Winpe

Les outils de base : procmon64.exe, procex64.exe, depends.exe, SPY++, Windbg, graphedit

Avec Winpe en 64 bits, il faut utiliser les versions 64 bits des logiciels (sauf si le sous-système 32 bits a été installé).

La version 64 bits des outils « sysinternals » est "intégrée" dans le fichier téléchargé sur Technet mais est invisible de prime abord. Elle est activée par la version 32 bits qui la rend visible pendant son utilisation.

- Comment isoler de manière simple ces versions 64 bits?
 - Modifier les paramètres de l'explorateur pour afficher les fichiers cachés.
 - Télécharger Promon/Procex depuis le site de Technet. Le décompresser.
 - Copier le programme Promon.exe sur le bureau.
 - Lancer Procmon. Cliquer que l'icone "arrêt capture"
 - Sur le bureau, il y a une icone de procmon64.exe (afficher les fichiers cachés). La recopier ailleurs avec, par exemple, le nom Procmon64.exe.
 - Modifier/désactiver l'attribut "fichier caché" de ce nouveau fichier « Procmon64.exe ».
- Bien comprendre les menus contextuels de Procmon :
 - "Stack" : fournit des informations sur la dll et le thread ayant déclenché l'action en cours. Cela peut orienter les recherches.
 - "Property" : permet de connaître la ligne de commande, bien utile et riche de renseignements.

Une observation directe avec Procmon

Un exemple pour fixer les idées : ajouter MMC.exe, la coquille vide qui lance les « snappins ».

- On lance Procmon et on lance mmc.exe.
- Rapidement un message s'affiche signalant une erreur.
- On arrête la capture.
- On ausculte la trace de Procmon : incompréhensible !

Mais on apprend vite.

Une erreur classique : trop de filtres. Ce qui manque se situe parfois, mais pas toujours, dans un autre processus appelé par le processus cible. Donc prudence avec les filtres.

Généralement, il faut aller vers la fin. Mais comme l'affichage inscrit lui aussi des informations dans la trace, il faut apprendre à les identifier pour arriver à la bonne information. Là, c'est l'expérience et la connaissance !

Et de proche en proche, de correction en correction (ajout d'une dll ou d'un programme identifié comme manquant, d'une clé d'objet COM, etc) on arrive à identifier tous les fichiers/clés manquants. Lorsque MMC.exe démarre sans erreur, on a donc la liste de tout ce qu'il faut ajouter, clés, fichiers (dont assembly).

Ensuite il faut mettre en oeuvre automatiquement ces modifications pour un prochain démarrage de winpe.

Là encore, plusieurs possibilités :

- injecter ces modifications dans Boot.wim,
- ou alors écrire un script Powershell pour injecter ces modifications lors du prochain démarrage.

💣 Les limites de l'investigation avec Procmon : les appels aux services, aux pilotes, aux objets COM, aux "named pipe", etc, ne sont pas tracés.

💣 Autres objets presque invisibles : les objets du système sont visibles avec WinObj.exe de « technet-sysinternals ». Voir mon script qui visualise ces objets.

Les symboles avec procmon

- dbghelp.dll et symsrv.dll

Il faut placer ces deux dll, dbghelp.dll et symsrv.dll, dans le même répertoire que procmon.exe. Ces deux dll proviennent de l'ADK/debugger. Avec un réseau opérationnel, la consultation de la « stack » devient plus explicite.

- cache local

Pour disposer d'un cache local (ici sur [g:\symcache](#)), il faut modifier une option de procmon, menu « Configure Symbols/symbols Path » comme ceci (sans d'espace) :

```
srv*g:\symcache*http://msdl.microsoft.com/download/symbols
```

Les symboles seront téléchargés depuis le site distant et mémorisés dans le répertoire de cache sur le disque afin d'accélérer les prochains affichages.

See : <https://developer.microsoft.com/en-us/windows/hardware/windows-driver-kit>

- Session System

Procmon affiche le message « loading symbol ... » mais le téléchargement n'a pas lieu et donc pas d'affichage.

C'est là que le cache local est très utile. On commence par faire les mêmes consultations avec la session ADM et le cache se remplit. Ensuite on recommence avec la session System : procmon consulte le cache précédemment rempli.

Stratégies d'investigation

La logique de la "fourmi" : des éléments unitaires

Pour chaque fonctionnalité, on recherche les éléments utiles un à un. Et on construit de grandes listes de clés et de fichiers pour chaque fonctionnalité.

La logique de "l'éléphant" : des éléments globaux

Partant du principe "qui peut le plus peut le moins", et ayant constaté par exemple que les clés de la ruche ClassRoot sont très utiles, on peut copier des pans entiers de registre sans se poser trop de question, tant que l'on comprend un peu ce que l'on fait.

Des morceaux de la ruche « Software »

Depuis une source bien choisie (une iso téléchargée et montée, un PC sous Win10 actif) on exporte des morceaux de ruche. Par exemple, CLSID, APPID...ou même ClasseRoot en entier.

La liste devient moins longue mais on perd un peu la connaissance du détail.

Le premier écueil est la perte de données lors de l'export car certaines clés sont protégées par des ACL.

- Méthode WinPeSe : avant toute chose, modifier les acl des ruches
- Identifier avec procmon les ACL générant une erreur

Il est possible aussi de faire des exports depuis Winpe en chargeant le fichier de la ruche à analyser.

Ensuite, il faut changer certaines références pour que Winpe continue de fonctionner :

- Pour "currentuser" si on prend des données dans la ruche de son PC actif :
remplacer c:\users\<noel> par x:\windows\system32\config\systemprofile
- Pour toutes les ruches :
remplacer "c:\" en "x:"
et "43,00,2A,00" par "58,00,2A,00" pour certaines clés de type binaire

La totalité de la ruche « Software »

On peut aussi prendre toute la ruche « Software » dans le fichier « install.Wim ». Il faut alors ne pas oublier de :

- supprimer les « runas » dans les clés « ...\classes\appid\ »
- recopier la clé « « ...windows Nt\winpe » depuis la ruche de WinPe dans la ruche destination provenant de « install.wim ».

Cela reste possible car la ruche software de install.wim build 10586 contient déjà des « X : », ce qui est très bizarre.

De plus, ne pas oublier que cette ruche contient aussi les clés pour le sous-système 32 bits.

La ruche « System »

On peut aussi se demander si on peut prendre la totalité de la ruche system dans install.Wim. Je ne retrouve pas le lien qui explique ces modifications.

Mais je sais qu'il faudra faire quelques modifications : modifier System\Setup\SetupPhase, Cmdline et désactiver certains pilotes.

La valeur de SetupPhase correspond aux phases décrites dans la documentation de l'ADK, Audit, Specialize... Elle oriente le comportement de Winpe, du logiciel Winlogon en particulier.

Ne pas oublier le rôle de la clé ProductPolicy.



la clé System\control...\control\MiniNt est un indicateur, ainsi que «SystemSetupInProgress», de l'activité d'un Os Winpe. Elle est testée par exemple par eventviewer.msc et powerhell « remote ».

La ruche « Drivers »

Faire le test pour vérifier si en utilisant cette ruche on peut alors éviter les ajouts de pilotes avec « dism ».

Je n'ai jamais fait ce test.

Les fichiers

On pourrait là aussi tout recopier mais cela fera un très très gros volume. Et ne pas oublier les fichiers de type « .mui » (liés aux langues disponibles). Les « assemblies » de « Dot Net », les pilotes, le « driverStore », seraient ainsi disponibles. Dans une Vm, cela peut s'envisager. Mais je n'ai jamais fait ce test.


Les comparaisons

Pour les fonctionnalités simples, il est parfois utiles de comparer les traces prises sur un Windows10 actif et sur Winpe. Par exemple, sur un Windows10 actif, avec le gestionnaire de tâches, on met fin à explorer. On active « Procmon ». Et on relance « Explorer.exe » depuis le gestionnaire de tâches. On fait de même avec Winpe et on compare. Cela peut aider parfois.

Une tentative de tutorial pour ajouter le bureau natif

Ouvrir mon script « traitement ». Localiser « fonction Etape3() ». Lire la séquence des actions à enchaîner. Les réaliser manuellement. Les listes des objets à traiter sont inclus dans le script.

Ce qui peut se résumer ainsi :

- Préparer le contexte : monter boot.wim et install.wim
- Charger les 6 ruches source et destination (software, system, default)
- Exporter les parties de registre : les variables \$ClesDeBaseSoftware et \$ClesDeBaseSystem contiennent les clés à exporter, modifier, importer.
- Modifier les Acl de la clé « OLE »
- Supprimer les runas de APPID (modifier les clés « APPID » nécessaires)
- Remplacer les C : par X :  Devenu inutile mais on ne sait jamais
- Ajouter quelques clés (voir fonction AjoutDeCles)
- Injecter ProductPolicy
- Démontier les ruches
- Copier les fichiers : la variable \$FileToCopy contient la liste des fichiers à copier
- Charger des pilotes souhaités

Donner la liste exhaustive des commandes à lancer ou des manipulation avec « regedit.exe » et « reg.exe » devient vite un travail gigantesque.

Un outil me paraît incontournable si on veut reproduire la création de son WinPe.

Partie Les scripts de MicroWinpeBuilder

Scripts de construction

Préambule

- La langue du WinPe construit : ce sera la langue de l'image ISO.
- Adk, ISO et host : version identique. Je n'ai pas testé dans une autre configuration.
- Je ne suis pas sûr qu'un fichier « install.esd » convienne.

Le fichier boot.wim utilisé est généré par l'ADK de windows10

Les scripts actuels utilisent l'arborescence de l'ADK pour WinPe. C'est donc dans le répertoire « Media » que l'on trouvera le fichier « boot.wim ».

Le script lance une seule fois la construction du fichier « boot.wim » en utilisant « cotype.cmd ».

Ce fichier contient tous les packages proposés par MS. Une fois construit par l'ADK, il est sauvegardé avec le nom suivant « boot.wim.AvecPaquetsDeBase.export ». C'est ce dernier fichier qui sera recopié en « boot.wim » afin d'être modifié par les scripts.



Il est possible de prendre un fichier boot.wim déjà créé par un autre outil (par exemple par WinBuilder) et de le modifier avec de nouveaux scripts (aucun intérêt de prendre les scripts actuels). Il faut pour cela le renommer en « boot.wim.AvecPaquetsDeBase.export ».

La machine host

Il faut utiliser une machine sous Windows10 build 10586. Divers programmes, comme DISM lors de l'ajout de pilotes, font des contrôles portant sur les fichier « .cat » qui pourraient ne pas fonctionner avec un autre OS.

Les deux sources de référence : ADK et Win10 Entr build 10586 ou plus

ADK

Il faut télécharger et installer l'ADK pour windows 10 build 10586 ou plus. Les scripts prennent en charge le lancement de « cotype.cmd ».

ISO de windows10 build 10586 ou plus

L'image ISO de windows10 build 10586 contient un fichier « install.wim ».

Il faut décompresser ce fichier « install.wim » dans un répertoire de référence.

A vous de modifier les scripts si vous le souhaitez pour automatiser cette tâche.

Description sommaire

GUI

Un premier script propose un GUI (ihm) rudimentaire pour saisir les trois chemins nécessaires :

- le répertoire qu'utilisera « cotype.cmd ». Il ne doit pas exister avant le lancement de « cotype ».

- le répertoire de base contenant l'ADK.
- le répertoire contenant la référence décompressée « install.wim ».

Ce GUI propose l'enregistrement de ces données dans un fichier « accolé » au script (flux alternatif).

Les traitements seront visibles dans l'onglet « console ». Un fichier de log sera produit en fin de traitement.

Tous les cas d'erreurs ne sont pas traités !

Traitement

C'est la partie principale la plus complexe. Elle assure les étapes suivantes :

- lancement de « cotype.cmd » pour construire l'arborescence du répertoire de base
- lancement de « dism » pour ajouter tous les packages de l'ADK : c'est très long !
- ajout de toutes les modifications identifiées (les miennes pour l'instant, les votre bientôt).

Les deux premières étapes sont réalisées une seule fois tant que le fichier « boot.wim.AvecPaquetsDeBase.export » existe. Les lancements ultérieurs du script utilisent la copie de ce fichier. Ils réalisent seulement la troisième étape.

Pour recommencer depuis le début : supprimer le répertoire de base de WinPe.

L'arborescence de base utilisée par les scripts

...\MicroWinpebuilder

C'est le répertoire dans lequel l'ADK vient déposer ses fichiers
contient les log et l'iso

...\MicroWinpebuilder\Add_Drivers

contient les pilotes pour mon imprimante USB

...\MicroWinpebuilder\Add_Root

contient procmon, winobj, etc

...\MicroWinpebuilder\MesProfilsWifi

contient les fichiers utilisés pour la connexion wifi (différents lieux géographiques)

...\MicroWinpebuilder\Media

généralisé par l'ADK

Les scripts ont pour but de modifier Boot.wim

...\MicroWinpebuilder\ISO

copie de ...\media pour construire le fichier iso

Ajout dans l'arborescence

- Drivers :

2 solutions, modifier le code du script « traitement.ps1 » ou « Dism add-driver » après avoir monter le fichier boot.wim

- Script dans la racine :

soit en modifiant les scripts « ihm » et « traitement » si le fichier est présent dans le même répertoire que le script « ihm », soit en le plaçant dans le répertoire « ...\\MicroWinpebuilder\\Add_Root »

- Imprimantes

Voir le script car chaque cas est unique

La structure d'un composant à ajouter

Pour un composant quelconque, on retrouve toujours les mêmes éléments :

- des fichiers
- des clés de registre
- des acl
- des traitements différés lors du démarrage de winpe (pour faire simple et rapide)

Pour optimiser les chargements/déchargements de registre et éviter un conflit avec « DISM », j'ai été conduit à éclater le traitement d'un composant. La logique du script devient :

- préparation des fichiers de post-traitement
- chargement des ruche
- pour chaque composant, modifications des registres
- déchargement les ruches
- Dism pour l'ajout de pilote
- pour chaque composant, copies des fichiers

Je reconnais que la modification ou l'ajout d'un composant devient vite compliqué dans ces conditions. Mais avec un peu de temps....

Les deux fichiers supplémentaires : ProductOptions et English

ProductOptions

Il contient les données relatives aux programmes autorisés dans Windows 10. Voir chapitre précédent.

English

C'est la traduction du mode d'emploi.

Script MonSetWow64 pour le sous-système 32 bits

L'origine

L'équipe WinPeSe propose un programme qui permet de mettre en œuvre le sous-système 32 bits. Ne voulant pas utiliser de programme externe, j'ai repris les sources écrites par un développeur chinois et je les ai converties en C#. Puis je les ai déposées dans le script pour les rendre lisibles.

Le principe

Lu sur internet :

yamingw found a solution for WoW64 in Win10PE

<http://bbs.wuyou.net/forum.php?mod=viewthread&tid=371490>

<http://winbuilder.cn/forum.php?mod=viewthread&tid=204>

Run setwow64. Ntoskrnl is phase 1 initialization testing if the WinPE is running in memory, it does not create a KnownDlls32 kernel objects. This object is populated with SMSS. When initializing a 32-bit system this object was not found in the path is wrong. This procedure only creates a path, no fill. Source refer to the <https://github.com/Google/SymbolicLink-testing-tools>, the CreateObjectDirectory and NativeSymlink merged. Kernel objects when the application exits disappear, so will both fill in Winmain.

SetWOW64 by yamingw. It works by creating the KnownDlls32 kernel object and linking it against X:\Windows\SysWow64 folder. It ~does what smss.exe should do. Note that it does not allow ThinApp packages to work.

Visualisation

Depuis un environnement 64 bits, on peut lancer ce script aussi bien dans WinPe que dans Windows10. Sans paramètre, il permet une visualisation des « objets system ». Il est rudimentaire et demande à être amélioré. Mais tel quel, lancé dans WinPe 64bits, il permet d'identifier d'éventuels objets manquants.

Note : winobj.exe n'existe pas en version 64bits.

Création

Avec le paramètre « create », il crée les objets nécessaires pour faire fonctionner le sous-système 32 bits.

Script de connexion WifiConnexion

J'ai repris un programme trouvé ici : <http://managedwifi.codeplex.com> . Je l'ai déposé dans un script PS avec un GUI. Il est fonctionnellement identique aux commandes « Netsh ».

J'ai rajouté la possibilité de déconnexion.

☀ le rafraîchissement avec ces API est assez long, parfois 30 secondes.

Script IHM_Get-Set_WinpeScreenResolution.ps1

Un copier/coller de code pris sur internet. Avec en fin de modification, l'envoi du message « WM_SETTINGCHANGE ».

Script de modification de FwFb.sys

Get-checksumPrg

Il permet de :

- contrôler l'information checksum d'un programme en comparant
 - le checksum inscrit dans le fichier par le linker
 - et le checksum calculé par une API de l'OS
- modifier ce checksum si besoin (param -modify true)

Il existe plusieurs API pour contrôler ces deux checksum. Je prends celle que j'ai déjà utilisée :

MapFileAndCheckSum de imagehlp.dll

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms680355\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms680355(v=vs.85).aspx)

La modification du checksum sera écrite dans le fichier du programme.

La description du format PE d'un fichier reste incontournable.

Sites :

https://fr.wikipedia.org/wiki/Portable_Executable

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms680336\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms680336(v=vs.85).aspx)

modify-octets

J'ai écrit rapidement ce bout de script. Je l'ai utilisé deux fois pour modifier le fichier fwfb.sys.

Il n'est pas optimisé. A vous de le terminer si besoin.

En l'état, il faut faire les deux modifications l'une après l'autre en mettant/retirant les commentaires.

```
# première modification
#[int[]]$lesOctetsAchercherEnHexa = ( 0x72, 0x07, 0x3d, 0x00, 0x04, 0x00, 0x00, 0x76, 0x02, 0x8b,
0xc3, 0xc1, 0xe0, 0x14, 0x89, 0x05, 0xa8, 0x46, 0x01, 0x00, 0x89, 0x05, 0xb2, 0x46, 0x01, 0x00 )
#[int[]]$lesNouveauxOctetsEnHexa = ( 0x72, 0x07, 0x3d, 0x00, 0xFF, 0x00, 0x00, 0x76, 0x02,
0x8b, 0xc3, 0xc1, 0xe0, 0x14, 0x89, 0x05, 0xa8, 0x46, 0x01, 0x00, 0x89, 0x05, 0xb2, 0x46, 0x01,
0x00 )
```

```
# deuxième modification
#[int[]]$lesOctetsAchercherEnHexa = ( 0xBA, 0x95, 0x00, 0x00, 0xC0, 0xEB, 0x22, 0x48, 0x25,
0x00, 0x00, 0x00, 0xC0, 0x48, 0x3D, 0x00, 0x00, 0x00, 0x40, 0x72, 0x11, 0xB8, 0x00, 0x00, 0x00,
0x20, 0x89, 0x05, 0x48, 0x50, 0x01, 0x00, 0x89, 0x05, 0x52, 0x50, 0x01, 0x00 )
#[int[]]$lesNouveauxOctetsEnHexa = ( 0xBA, 0x95, 0x00, 0x00, 0xC0, 0xEB, 0x22, 0x48, 0x25,
0x00, 0x00, 0x00, 0xC0, 0x48, 0x3D, 0x00, 0x00, 0x00, 0x40, 0xEB, 0x11, 0xB8, 0x00, 0x00, 0x00,
0x20, 0x89, 0x05, 0x48, 0x50, 0x01, 0x00, 0x89, 0x05, 0x52, 0x50, 0x01, 0x00 )
```

Une recherche octet par octet en PS prend du temps .La recherche en Ps reste possible pour un fichier de 100 ko.

Le nom du fichier à modifier est figé dans la variable :

```
$filePath = "C:\Users\noe\l\Desktop\informatique\fbwf\modif-en-cours-fbwf.sys"
```

A vous de l'adapter à votre besoin.

MicroWinpeBuilder

Learn how to adapt oneself a Winpe10 64 bit by adding, for example, MMC (devices, disks, services, firewall), the office with the taskbar, ncpa, netsh, wifi, MSI, the files server, audio, wow64, session Adm, bits, winrm, printer/scan usb/network...

A set of scripts written in Powershell illustrate this learning.

This document will be possibly useful for curious beginners like me. I doubt that an expert is one any interest.

It is now for you to complete and correct this document.

Presentation of the aim pursued

- Understand how to adapt Winpe10 generated by the ADK
- Learn how to make basic investigations
- Integrate information discovered or made available on internet
- Attempt to provide additional for the beginner.

Warning

Scripts proposed here can build a WinPe. But due to the technology of the scripts PS based on DotNet, the size of the Boot.wim file is above 600 MB. More, powershell startup is slow.

It is a building for educational purpose.

The constraints

Since it is to learn and understand, everything must be visible.

- Rule 1: do not use external software (i.e. without «Winbuilder»)
- Rule 2: only use Powershell scripts (including the `c#` if necessary)

The principle

I collected a set of information in various sources available on the internet (like Kullenen_Ask: <http://www.msfn.org/board/topic/143241-portable-windows-7-build-from-winpe-30> and a few others) and mainly on TheOven (<http://theoven.org>) site.

Learning is to read and understand this information. Then implement to build « his »WinPe.

To illustrate this learning and show the relevance of this information, I've written a few PS scripts. They provide a rudimentary WinPe but that works. They use as input the « boot.wim » file created by the ADK. So you can add your drivers with DISM at any time.

Now, to investigate and to integrate you need, create your own scripts, edit the PS scripts supplied with MicroWinpeBuilder...

Investigation and implementation of the results

The investigation led to identify objects (files, registry keys, ACL) need to inject into the "boot.wim" of Winpe.

It allows to:

- build lists of items to add
- adapt scripts according to his desire and his discoveries.

Before, I manually fixed the results of my investigations. Now, the PS scripts inject changes during the development. And the same scripts also enable to build the final WinPe.

And the most important is that the initial file is always the same, the boot.wim file created by the AIK. This is what ensures the reproducibility of tests.

A quick comparison with the reference WinPeSE

WinPeSe uses the Win10Enterprise Evaluation ISO file. It does not use ADK (except a few tools that are downloaded directly if I understood).

This ISO file contains two essential files:

- Boot.wim, the MS WinPe used during installation of Windows 10.
- Install.wim, which is the image of services10

WinPeSe captures so these two files. It extracts the necessary files for which lists are contained in Winbuilder scripts. The hives are copied and modified. All these elements are added to the 'target'. WinPeSe also brings its own files and programs in its scripts (binary code encoded on 64-bit?). Then Boot.wim is assembled.

It is the precision and great art. But before that he had to find all changes to treat. And there is yet more complicated.

The automation of the program hides the complexity of the preliminary work and the knowledge necessary to build this WinPe. WinPeSe scripts are of course rich in detail and carry on items on the more obscure and complex lighting.

My only remark: it was impossible to modify subsequently the file Boot.wim of WinpeSe with DISM to add a package of ADK as dotnet.

Sites :

<http://wb.paraglidernc.com/Help/default.html> and <http://www.paraglidernc.com/winbuilder/default.htm>

What is assumed known

- having read the WinPe documentation available on MS sites
- be familiar with the tools available in the ADK and know how to use DISM, BCDEDIT, BCDBOOT ...
- learn to change a BCD
 - And don't forgot that the BCD for WinPe contains the keys for the RamDisk which are not present in the BCD of the hard disk after a normal installation of windows 10
- learn to build the BCD to a VHD.
 - The VHD are very practice for investigations because the deposited files persist after the reboot. Changes to the hives do not persist.
- reading scripts CopyPe.cmd and MakeWinpeMedia.cmd of the ADK

Structure of the document

A first part gathers all necessary informations to WinPe.

The second part gives some ideas to launch the investigation.

The third provides an illustration, an implementation by scripts.

Too much detail kills information

This document doesn't describe all the details (keys, files, acl) present in the PS scripts.

See and read the scripts if you want to check the presence of a file, a key...

Part knowledge of WinPe

The change of the hives : « C : » replaced by « X : »

Different sites indicate that this change (text and binary !) is required. But since at least build 10586, my scripts do not conduct this replacement and it works.

Least know about desktop Explorer.exe

Not very nice but sufficient

Get a desktop with a certain comfort in the handling of windows need to implement the Window Manager "Dwm.exe". Without "Dwm" and with Ribbon, it appears a disruptive black band above the Ribbon.

"DWM" requires the CoreMessagingRegistrar service for build 10240 and also the driver WindowsTrustedRt for build 10586 (see Win10PeSe).

How did the team WinPeSe found that? I do not know!

With an active Winpe, the "Default" hive in regedit corresponds to the file '..\config\default'. It is used by the « System » account to become "HKCU ». Various keys used by "explore" should be injected for operation without anomalies.

...\Software\Microsoft\Windows\CurrentVersion\Explorer\UserSignedIn=1 avoid the delay before the appearance of icons in the task bar.

Without the 'Themes' service, I have not managed to display a background image to screen after « explorer.exe » starts. This service requires DWM.

Colored « title bar» for the active window:

the following key appears sufficient: HKLM\...\DWM\ColorPrevalence = 1

«WallPaper» display is still a bit obscure for me.

The minimum about Explorer.exe

"Explorer.exe" performs two different features:

- It created an desktop for the user who opens a session
- It allows to explore the files system in the PC

The second feature is available in an original Winpe. The « explorer.exe » file is missing but the feature is active. To check this, in an active original Winpe, open "Notepad", menu "file/open" and it gets a File Explorer that allows almost all actions on the file system. Its implementation is uncomfortable without a desktop.

The first feature, desktop, requires the file « explorer.exe ». If we start this program from a box "Cmd" and by plotting with "procmon", one can find the missing elements.

Then, it is necessary that Winpe automatically launches "explore" at startup. For that, you need add the key "...\windows Nt\winlogon\shell = explorer.exe". When "explorer.exe" is launched for the first

time, it tests for the presence of this key and installs the user's desktop. At the second launch, it opens a window to explore the file system.

"Explorer" is implementing a set of elements COM The COM dialog uses clients and COM servers. Dialogue COM implements permissions depending on the configuration of the COM components defined in the registry. To "explore", many COM components are configured to request permission. But Winpe (WinRe?) does not implement mechanism to respond to these requests for permission.

To Bypass this security, you must modify the « **HKCR\APPID\RunAs** » values for COM objects (all preferably).

This point remains mysterious. But if I understand correctly, the absence of the "runas" parameter indicates to the COM server that it has no need to check the identity of the logged on user. If this parameter is present (and equal to InterActive), the server checks the user's identity. However, this control fails with WinPe as the 'system' account is not an ordinary account.

To learn more about security in COM :

[https://msdn.microsoft.com/en-us/library/ms682359\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms682359(v=vs.85).aspx)

[https://msdn.microsoft.com/en-us/library/ms680046\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms680046(v=vs.85).aspx)

The function « desktop display »

The discovery late source: <http://theoven.org/index.php?topic=1722.msg20025#msg20025>

I should have started by now. But this is, how do I know? Where the interest of an effort of centralization of the documentation, well that changes are very fast.

The activation of this function:

- with the keys "Windows" + D
- by clicking in the lower corner on the right (1 mm wide) between the clock and the edge of the screen
- using the context menu from the task bar

In Winpe:

The window "Shell_TrayWnd" receives messages (WM_XXXX) sent by various sources of activation but not treated. "

A bit of investigation with SPY ++

Spy ++ (essential and present in visual studio community free) shows the following messages received by "Shell_TrayWnd":

Msg 0 x 579 wParam lParam 3 0

Msg 0x43F

In a Windows 10: sending the msg 0 x 579 (wParam lParam 3 0) is enough to switch the display of the desktop.

Idea: try to understand why 'explorer' does not process these messages using windbg.

Fixed in Winpese:

The Wind.exe program and the dll MsgHook.dll provides treatment expected.

Also simplified these files: basically, when receiving the right message in the hook, shrink or enlarge the main windows for the desktop display or return to the previous display.

Note: how is this Wind.exe/Myhook.dll program launched in WinpeSe?

The « users\default\ntuser.dat » file is used by the session « system » . It is also copied to be used by the adm session.

This file runonce key contains multiple values, allowing the launch of programs.

And value "autorun = autorun.exe" to process the file autorun.ini containing orders to launch, including "Wind.exe".

Changing the size of the screen

When you change a parameter as the size of the screen, the program responsible for this change sends a message 'WM_DISPLAYCHANGE' to all main windows. And each acts accordingly and resizes. But in Winpe, it seems that Explorer does not process this message. But it treats the message "WM_SETTINGCHANGE" which also leads to resize the window of the office, and so all the other windows.

Solution for my script to change the size of the screen:

send "WM_SETTINGCHANGE" message at the end of the screen resizing

Solution of WinpeSe with "FixScreen (JFX)" and Broadcast EnvChange (Galapo) programs":

loop: wait for the message "WM_DISPLAYCHANGE" and send the message 'WM_SETTINGCHANGE.

My script to change the size of the screen: IHM_Get - Set_WinpeScreenResolution.ps1

To read the video configuration in PS:

```
Add-Type - AssemblyName system.windows.forms
[System.Windows.Forms.Screen]: allscreens
```

About abnormalities

Abnormalities corrected in my winpe, for memory!

- Wpedit hangs 5 minutes at startup of winpe
 - "The 'policymanager.dll' key was present but the key software\micros...\policymanager' was absent.
- Starting the service 'coremessagingregistrar' failed

it lacked the key 'software\micros...\securitymanager '.

- The desktop icons were displayed one minute after the office missing keys in "HKCU\..\explorer.
- The sound did not work and the notification icon at the bottom right displayed 'no speakers '.
an ACL prohibited access to the key "...MmDevices\Audio\Render\...\properties" to the accounts used by the AudioSrv service.
- Cannot move icons on the desktop

It came from the software\microsoft\ole key that did not contain the DragAndDrop-related information. The new load failed due to the Acl of the key.

- Creation of impossible shortcut on the desktop
it lacked the "appwiz.cpl" and "osbaseln.dll" files as well as their ".mui.
- No background screen: missing productOptions\productPolicy and various keys
- snippingtools did not work: missing productOptions\productPolicy
- Right click on the wallpaper «display, personnalization» launches nothing: requires the slot 32-bit system
- BITS and Powershell : OK in session adm
- Mstsc : Ok with NLA

Abnormalities emerged with the version 1607 build 14393

- Opening of session 'adm', a logonui.exe error message appears but is not blocking
- WMP does not work
- In the case of a VHD, the Adm session does not open and hang up (possible workaround)

Persistent anomalies

- In the case of starting on a USB key, and at the opening of the session "adm", the launch of a program generates a warning "Publisher unverified".
- first launch of Powershell very long: .cat pb
If I copy all the files cat from install.wim then wait one minute before powershell becomes operational. With Procmon, found that a svchost creates the file "...cartoot2\...\catdb".
- impossible to pin to taskbar right-clicking
- the "eject USB" icon does not eject because it lacks an entry in the context menu. This did not very serious because there is no cache write on USB keys.
- Remote powershell and Wsman?
- network with netsh trace: ndiscap.sys starts, the ETL file is generated but not the CAB file

- ...

DCOM and the toggle to the classic GUI

The new graphical user interface called 'Metro' seems impossible to implement in Winpe. However some actions are defined by the value of a pointer in the registry. It is sometimes possible to modify this pointer and activate the classic interface of a component.

A comparison with windows7 to identify the differences relating to the new interface.

For example, the right click on the desktop to select items "Display settings" or "Customize" from the control panel. This triggers the display of the new interface. A judicious change can find the old interface.

New interface 'metro':

```
HKCR\DesktopBackground\Shell\Display\command\DelegateExecute = {556FF0D6-A1EE-49E5-9FA4-90AE116AD744}
```

which activates the COM object: CLSID_LaunchSettingsPageHandler

Back to the old interface:

```
HKCR\DesktopBackground\Shell\Display\command\DelegateExecute = {06622D85-6856-4460-8DE1-A81921B41C4B}
```

which activates the COM object: COpenControlPanel

But in the case of the display, one is faced with the behavior described in the following chapter.

DCOM on 64-bit: a bug with Explorer?

In Winpe 64-bit, if it launches "Desk.cpl" for example, task manager shows that the dllhost.exe software is loaded. This program is a 'surrogate' part of the DCOM architecture. No display takes place. A new launch of «Desk.cpl» loads a new process "dllhost.exe". Dialogue DCOM client-server seems impossible.

Under the same conditions with a normal Windows 64 bit, "Procmon" shows that there is «x:\windows\syswow64\dllhost.dll» loading then quickly unloaded. This is the 32-bit version of 'dllhost'.

However, in the classic version of the ADK Winpe64, the 32-bit subsystem is absent. It's the source of the anomaly.

Is it possible to 'Explorer' to load a Server DCOM 64 bits of this feature? I don't know.

Sites to visit:

Security in COM : [https://msdn.microsoft.com/en-us/library/ms693319\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms693319(v=vs.85).aspx)

dllhost : [https://msdn.microsoft.com/en-us/library/ms695225\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms695225(v=vs.85).aspx)

APPID : [https://msdn.microsoft.com/en-us/library/ms678477\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms678477(v=vs.85).aspx)

<https://blogs.msdn.microsoft.com/jigarme/2007/10/09/what-is-appid/>

Com-32-64 : <http://mariusbancila.ro/blog/2010/11/04/32-bit-and-64-bit-com-servers/>

WOW64: the 32-bit subsystem in Winpe

WinPeSe offers. It is therefore possible.

A Chinese site (but I forgot which) explains that Winlogon creates normally 2 objects system for the operation of the 32-bit subsystem.

A Chinese developer wrote the software «SetWow64.exe». This software is taken up by the WinPeSe team.

Sub-systeme32 also requires to enrich the 'WinSXS' directory.

What are system objects?

See MSDN and these sites.

In french:

<http://www.ivanlef0u.TuxFamily.org/?p=39>

Object Directories:

[https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff557755\(v=vs.85\).aspx](https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff557755(v=vs.85).aspx)

NtCreateDirectoryObject

[https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff556456\(v=vs.85\).aspx](https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff556456(v=vs.85).aspx)

ZwCreateDirectoryObject routine

[https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff566421\(v=vs.85\).asp](https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff566421(v=vs.85).asp)

To explore the system objects: 'winobj.exe'

Need to download it from the 'technet/sysinternals' site. It is 32-bit only. Therefore, you cannot use it with a normal 64-bit WinPe.

It is also for this gap that I wrote the PS script.

The 'MonSetWow64.PS1' script

I replaced the "SetWow64.exe" program by a PS (rule 1) 'MonSetWow64.PS1' script. It includes two features:

- Launched without parameters, it allows to visualize objects in the system (like winobj) in a 64-bit system.
- With the 'create' parameter, it creates 2 necessary objects.

This script is usable even if there is no 32-bit subsystem. It had long wanted a way to visualize objects in the system in 'full 64-bit' but I didn't 'NTxxx, ZWxxx' API.

I know that this script still requires work.

With active Wow64, the previous DCOM (previous chapter) dialog uses the 32-bit subsystem because "Desk.cpl" works and trace "procmon" confirms the launch of the 32-bit version of 'dllhost.dll '.

ProductPolicy: WallPaper, SnippingTool.exe...

The key

'HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\ProductOptions\ProductPolicy' contains the information necessary to enable certain features or components as wallpaper, program «snippingtool.exe», etc.

Its contents change with the versions of Windows and installed components.

This key is exported from a build 10586 assets services10 because I do not know how to do otherwise.

I would have never found this point without WINPESE. The comparison has been long.

There is a software on the net that allows to view. It allows to modify this key on offline mode.

<http://reboot.Pro/topic/20585-productpolicy-Viewer/>

<http://www.remkoweitjnen.nl/blog/2010/06/15/having-fun-with-windows-licensing/>

Changing the wallpaper and themecpl.dll

The context

The first step was to make possible the access to the menu « Personalization ». Thanks to those who have found the mechanism of change of GUIDs in the key "... \DesktopBackground".

This menu is accessible from a right click on the desktop.

You can open the « Personalization » menu with the command :

```
control.exe /name Microsoft.Personalization
```

The « Personalization » window (bottom) offers a « Desktop Background » link that allows you to change the wallpaper.

You can open this window with the command:

```
control.exe /name Microsoft.Personalization/page pageWallpaper
```

You can also open the window « Color » with:

```
control.exe /name Microsoft.Personalization/page pageColorization
```

Sites:

a gateway but we must go back in the tree view of the site:

[https://msdn.Microsoft.com/en-us/library/bb773213\(VS.85\).aspx](https://msdn.Microsoft.com/en-us/library/bb773213(VS.85).aspx)

Amendment of shell32.dll:

<http://superuser.com/questions/494878/does-Windows-8-have-a-status-bar-to-display-details-of-a-file-idem>

<http://sumtips.com/2012/11/Windows-8-move-details-pane-to-bottom-of-Explorer-window.html>

various

<http://vistastylebuilder.com/forum/index.php?topic=215.0>

First pitfall

But if attempts to open this window by clicking on the link « Desktop Background » we get the error : « ms-setting:personalization-background: the specified module is not found ». Indeed, this link points to an application of the Metro interface unusable in WinPe.

Question: how to make active this link?

After some research, opening the file themecpl.dll with notepad, found text strings that resemble scripts in xml. They contain the words "duixml", "shellexecute", "pageWallpaper" and "pageColorization". By using a software exploring the resources of a program, there are scripts in resources. A reading to quickly get an idea of the logic.

The « Sounds » GUI is started with the command:

```
"shellexecute="%windir%\system32\control.exe"shellexecuteparams = " / Microsoft.Sound/page 2
"name
```

The GUI of the screen saver is started with:

```
"shellexecute=%windir%\system32\rundll32.exe" shellexecuteparams = "shell32.dll,
Control_RunDLL desk.cpl,ScreenSaver,@ScreenSaver"
```

Quickly is how to change the text to run the two commands that interests us.

The only resource to edit: « UIFILE/1001 ».

The two strings of text to modify:

Before	ShellExecute = "ms-settings:personalization - background"
After	"shellexecute="%windir%\system32\control.exe"shellexecuteparams = " / Microsoft.Personalization/pageWallpaper page name "

And

Before	ShellExecute = "ms-settings:personalization - colors"
After	"shellexecute="%windir%\system32\control.exe"shellexecuteparams = " / Microsoft.Personalization/pageColorization page name "

The tool to manipulate resources, explore and modify (big green compilation button) and save changes: ResourceHacker here :

<http://www.angusj.com/resourcehacker/>

Just replace the original file in WinPe (mui file is unchanged).

Then we test this new file in WinPe.

Second pitfall

The second pitfall is more difficult to describe without screenshot. But this too lengthen the PDF.

The « picture location » combobox contains multiple entries. They are ineffective in my context.

Only « Solid Colors » is active. I do not know why.

A solution:

- During the construction of winpe, I deposit the "wallpaper" pictures in a directory at the root of X, or x:\fleurs in my case. As I use a VM, this becomes very simple. For test without VM, to mount the boot.wim with Dism for example and copy the directory to the root. Then unmount.
- On the desktop of WinPe, I open the menu « Desktop Background » (right click on the wallpaper and enable the "Desktop Background" link or launch the good control command, see above).
- Click on the button « Browse... ».
- Select the directory containing the images. Either x:\fleurs in my case.
- In the combobox, select the entry for the directory. And the images appear.

Then, you can change the wallpaper directly.


I hope that someone will find why the initial wallpaper does not appear, nor even its corresponding entry in the combobox.


Cannot edit resource in PS? Is it possible to compile c# resources, with csc.exe for example?

IEFull64 : nearly OK

Info: IE requires key:

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings

 If the hku\default\...\cache\persistent value (dw 1) is absent, then IE Launches 3 rundll32 of removing the traces of the user.

 Note the presence of a test program: "X:\Program Files\Internet Explorer\iediagcmd.exe". It uses DxDiag, nesth, ipconfig... The result file seems to be complete.

Key x86AppPath

I make three tests :

- First test: the 32-bit subsystem is active
 - Launch of iexplore 64-bit : fugitive display
 - Fill x86 of internet directory explorer: a window appears and procmon reports that 32-bit iexplore restarts continuously.
 - Launch of 32-bit iexplore: display error message "0xc0000034 cannot start.
- Second test: subsystem 32 inactive bits but the Directory x86 of internet explorer is present
 - Launch of iexplore 64-bit : 64-bit version try to run the 32-bit version. Displaying a window.
- Third test: I run 32-bit iexplore: display a window but cursor "current activity".

First search with procmon:

Iexplore x 64 consults the key : hklm\software\microsoft\internet explorer\main\

x86AppPath = x:\program files (X 86) \internet explorer\iexplore.exe

Then « createProcess » of this 32-bit version with the parameters

« scodef:2760 credat:xxxx /prefech:2 » where « scodef » is the Pid of the process iexplore 64 bits.

- New tests:
 - delete this value: fugitive display
 - make pointer «x86AppPath» to the 64-bit version: we get to see several "IE" in the task bar and view the contents of "MSN".

Utility of Loosely-Coupled IE (LCIE) ?

First discovery on internet:

<http://blog.HttpWatch.com/2009/04/07/seven-things-you-should-known-about-IE-8/>

« TabProcGrowth = 0 is a registry edit that will disable the Loosely-Coupled IE (LCIE) function in Internet Explorer 8. Essentially what this means is that all tabs in Internet Explorer will be handled by one process of iexplore.exe. This also means that Protected Mode will be Off and that if one of your tabs crash, Internet Explorer will crash. »

There are more instances of "iexplore" with the parameter "scodef". To use the task bar and icons of IE to move from one tab to another.

The main window of IE containing menu is now displayed

I did the test with the build 14393.Chance one more...The IeFrame.dll.mui file was simply missing.

F12 (debugger IE)

Many things OK. Capture network is ok.

Fiddler and https

During my test, Fiddler ask for a certificat and propose a link for downloading it. I install it with :

```
certutil.exe -f -user -addstore root <fichier_certificat>
```

And after thath, i can see network trace in « fddler ».

Anomalies identified

Session "System":

- Some menus do not work
- Download: files are well downloaded to a cache local directory but are not displayed

Session 'ADM ':

- The management of passwords does not work
- The downloads work as well as the display of downloads.



Be carefull, if **date/time** is not good, some sites reject the authentication request.

PowerShell: very slow start

The first start of Powershell is still very long.

In the site <http://theoven.org/index.php?topic=1639> forum: "sezz" explains:

I also figured out why starting PowerShell in Windows PE is extremely slow (first start takes about 20 seconds for me):

- NGEN hasn't run yet -> the native image cache doesn't exist
- CATDB hasn't been generated yet

Running "NGEN.EXE update /NoDependencies /Silent" takes ages, so I ran it once and fetched the files from "X:\Windows\assembly", now I theoretically could use them everytime when building an image, but they are very large (about 300MB)...

The CATDB gets created on the first start of PowerShell.exe, that's why it takes so long. The file "X:\WINDOWS\SYSTEM32\catroot2\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\catdb" can be backupper (stop CryptSVC service first) and then used when building an image. It's only about 20MB.

Couldn't find a solution yet without running Win PE first and grabbing the CATDB, but it's good enough for now :)

Change of name of the "computer".

Two names, one for the machine and one for the network.

WinPE generates a random name for the « computername ».

The WinPeSe script indicates a need to modify the following key to impose a predefined name:

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\WinPe
```

```
SetComputerName = 0 type Reg_DWORD
```

The question is: how has identified this key? Do the test without this key!

Then the name of the 'computer' must be registered in the following keys:

```
...\SYSTEM\ControlSet001\Control\ComputerName\ActiveComputerName
    ComputerName = « MyComputerName »
...\SYSTEM\ControlSet001\Control\ComputerName\ComputerName
    ComputerName = « MyComputerName »
...\SYSTEM\ControlSet001\Control\services\TcpIp\Parameters
    HostName = « MyComputerName »
...\SYSTEM\ControlSet001\Control\services\TcpIp\Parameters
    «NV HostName » = « MyComputerName »
```

Login with the administrator account.

Without the work of the WinPeSe team, I would have never known to implement user login with the administrator account.

Native sessions in WinPe:

- Session 0: for services
- Session 1: automatically created by WinPe when starting on behalf of "system".
- Session 2: it will be created at the opening of the session by the "adm" account.

In this session 'adm', the 'administrator' account has full rights as it belongs to the Administrators group. But these rights are not active!

The general principle read in WinPeSe

At its start, Winpe enables the session for user 'system'.

We prepare the automatic logon keys: 'mechanism AutoAdminLogon '.

It must absolutely stop and change the "start = disabled" configuration of two services Gpsvc TrustedInstaller.

Since the session 'system', we disconnects the console of this session with "tsdiscon.exe"

The session 'system' is not closed. But the ' keyboard/screen/resources' console is no longer assigned to this session.

WinPE understand that the ' keyboard/screen/resources' console becomes available. It therefore

proposes to another user to open a session with the GUI of "logon".

However, the "AutoAdminLogon" mechanism has been previously configured. Therefore Winpe has authentication components of the new user (workgroup, account, password)

The logon mechanism controls the identity of the account 'adm '. Then creates the profile for this user directory.

It therefore has two interactive sessions, one for the 'system' account for the 'administrator' account.

We go from one to the other with the command "tscon", "tscon 1" to activate the session 'system' and 'tscon 2' to activate the session 'administrator'.



To avoid the trap of empty password, I change the password of the administrator. You must change the password in the script !

«Computer» must belong to a 'workgroup' :

Make sure that the 'computer' has joined a "workgroup" to avoid later having the error message "inaccessible domain or workgroup".

If we force the 4-key value defines the name of the computer, then the computer is not registered automatically in the «workgroup».

To join a "workgroup", you can use the API "joinDomainOrWorkgroup" of WMI for example.



Vbs script to test in WinPeSe.

```
strComputer = "."
Set oWmiService = GetObject("winmgmts:\\.\root\CimV2")
Set colComps = oWmiService.ExecQuery ( "Select * from Win32_ComputerSystem" )
For Each oComp in colComps
  wscript.echo "oComp.workgroup"
  res = oComp.JoinDomainOrWorkgroup("Workgroup")
  wscript.echo res
Next
```



Script in Ps for MicroWinpe.

```
$s = gwmi win32_computersystem
$s.JoinDomainOrWorkgroup('WorkgroupWinpe')
```

🔊 Surprise:

With a right click on "Properties" of "This PC", there is:

"Working Group: not available"

Then if you click "Edit settings" then "Computer Name", you will see:

"workgroup = * Unknown *"

? Point to verify:

If it does not set the value of the 4 key defines the name of the computer, it seems that "wpeinit" automatically ensures the integration of the computer into the "workgroup"

Slowdown at the opening : corrected

There are many sources of slowdown in user login 'administrator'. I identified these:

- Display the message « Please Wait » for 10 minutes

This occurs if two GpSvc and/or TrustedInstaller services are not configured with « start = demand »

- the network is mounted before the opening

It is wpeinit rising network. With procmon, I note that winlogon is attempting to use the resource:

\\LsaSetupDomain*\MAILSLOT\NET\NETLOGON

A time-out of approximately 6 seconds seems active. And the call is revived several times.

I did two tests:

- not the VM network adapter
- presence of the card but not the wpeinit program

In both cases: the mailslot writes do not because the mailslot is "disconnected" but there is no slowdown.


- ntuser.dat does not contain "UserSignedIn"

The absence of the 'hkcu\...\explorer\UserSignedIn' key introduces a delay of almost one minute after the display of 'Preparation '.... ».

- service 'SENS' I put in place for BITS introduced a further delay of two minutes. Winlogon sends notifications to various programs and waits for their response. In my case, the 'SENS' service did not work properly.

 Winlogon notifications are visible in this key must be changed:

"... \ControlSet001\Control\Winlogon\Notifications\Components\"

 Surprise : « LsaSetupDomain » is also visible in the SMB trames sended on the LAN during the connection to a printer for example.

Error 0x80000003: several cases since the build 14393

In the case of the opening of the session "Administrator", it appears the error 0x80000003. It's not blocking. Simply click on the « OK » button and log well held.

In the case of the « switch user », the attempt to open a new session hangs the system.

In the case of a boot from a VHD "flat" (from a VM hyperV or from USB due disk) mode, the session "Administrator" does not open.

Chance has done things: a site proposed for download a file of 4 GB with a version of Winpese and a set of applications installed (mostly portable applications not requiring their integration research). And here, everything works. See <http://theoven.org/index.php?topic=1873.0>. Allowed to find an effective solution.

Solution:

It is the Imageres.dll file that is involved. It is smaller than that of the ISO reference.

Another possible workaround (for memory):

Rename « Windows.UI.Logon.dll » to « Windows.UI.Logon.dl_ »

In the current state of the scripts, it is possible to create a file x:\NoAdm which prohibits the launch of "tsdiscon.exe". We can do all the necessary investigations before to launch tsdiscon manually .

Deactivation of the graphical part of logonUI

It is possible to disable graphic party «logonUI. Just rename the file « windows. UI.logon.dll » in « windows.UI.logon.dl_ ».

Part logonUI in mode console («ConsoleLogon.dll») shows a black box and shows the same texts but without animations.

Another possibility: replace logonUI.exe by cmd.exe

This allowed me to draw with "procmon", well that without result. But the method may serve me someday, may be:

- Rename logonUi.exe in Org-logonUi.exe
- Copy cmd.exe in logonUi.cmd
- Prepare the keys and commands needed for logon with administrator
- Run procmon
- Launch the logon procedure: tsdiscon
- CMD takes the hand. Start a powershell. Recover with gwmi command from CMD line.
Launch then Org-logonUI.exe with the parameters of the line «cmd»

On failure, Org- logonUI loop. Can be a 'CTRL-C' to get out.

So back to the CMD. And with ALT - TAB, we return to powershell. Can then return to the session 'System' with tscon 1. And stop Procmon.

Runas only from ADM session

Can check the functioning thus:

```
NET user MonToto MyPassword/add
```

```
net localgroup administrators/add MonToto
runas /noprofile / user: MonToto cmd.exe
and then type the password "MyPassword".
```

Do not forget the parameter "/noprofile" otherwise there is the error "the device is not ready.

This command runas doesn't work in System Session

The warning « the Publisher could not be verified »

When Winpe boots from a USB key or in a VM with an iso file, all the programs launched by "Windows + R" trigger the display of this warning. But if decompressing the boot.wim file in a VHD (with a suitable BCD), this warning does not appear.

I see that the only program that does not trigger this warning is: tscon.exe

Workaround: at the onset of this warning, switch in the session "system" with "tscon.exe 1. Then return in the session 'adm' with ' tscon 2. Current powershell program and its descendants will still trigger the warning but not new programs.

 Surprise: with the feature "**Open Command Here**", the launch of the cmd.exe program does not trigger this warning.

Site describing the 'Open Command Here' feature (see option 8 on the page):

<http://www.tenforums.com/tutorials/3288-command-prompt-open-Windows-10-a.html>

To be implemented in the context menu without using the Shift key:

<http://www.askvg.com/enable-Open-command-window-here-option-in-context-menu-in-Windows-Vista>

Delete the values:

HKEY_CLASSES_ROOT\Directory\Background\shell\cmd\extended

HKEY_CLASSES_ROOT\Directory\shell\cmd\extended

The network with netsh trace

[https://technet.microsoft.com/en-us/library/dd878517\(v=ws.10\).aspx#bkmk_traceStart](https://technet.microsoft.com/en-us/library/dd878517(v=ws.10).aspx#bkmk_traceStart)

Currently:

- It is still incomplete
- the PLA service is mandatory for a "good enough" operation of the command "stop".
- This PLA service requires the modification of the ACLs for the key 'service\pla\configuration '.
the 'stop' command does not generate the expected file set (no file .cab...)

- the ETL file fills only if the firewall allows the "inbound connections" or an exception is created

The collection of information system requires service "Schedule". But it blocks copying files in a VM. So I will not start.

Much research for a small gain, the firewall log contains a better summary of the frames in the network.

Using providers currently gives better results than the use of scenarios.

The ndscap.sys driver installation

Files used: ndiscap.inf, ndiscap.sys and the .cat identified with « signtool ».

The ".cat" file is it really useful?

The installation of the driver in the stack 'network' is performed after the «Winpe»:

```
netcfg.exe -c -p -i MS_NDISCAP
```

The parameter "-e" of netcfg.exe prohibits the installation of the driver. Incomprehensible!

It requires the file 'ndiscapfcg.dll '.

Startup succeeds: net start ndiscap

The capture is possible with a Wifi or Ethernet card. OK in a VM.

Configuring with netsh

It should be added in the 'system' hive «netdiagfx» and «Nettrace» keys that contain the scenarios available.

These two keys are protected by ACLs.

- To start a capture with a provider:

```
netsh trace start provider = Microsoft-Windows-TCPIP capture = yes report = yes correlation = yes overwrite = yes level = 5
```

- To start a capture with a scenario:

```
netsh trace start scenario = InternetClient capture = yes report = yes
```

- To stop a capture:

```
netsh trace stop
```

The "netsh trace stop" command does not generate the file '.!. Cab» or the «report» file

- To convert the ETL in TXT file:

```
netsh trace convert input = <chemin>\NetTrace.etl dump = TXT
```


The 'txt' file contains all information expected but TCP traffic is present.

See : <https://isc.sans.edu//diary/No+Wireshark?%2bNo%2bTCPDump%3f%2bNo%2bProblem!/19409>

Must see: <https://www.microsoft.com/en-us/download/details.aspx?id=44226>

The "netsh trace stop" command does not generate the file '. ' . Cab» or the «report» file.

The event log

Don't forget to add the key '... \services\eventlog\applications' and '... \services\eventlog\system'.

'Eventlog.msc' displays of the logs. To do this, it must implement the following sequence:

- stop the eventlog service
- Rename the key MiniNt
- restart the eventlog service

Oddly, it happens that the events are more registered.

The audits

The activation of certain audits allows to enrich the event "Security" log.

AuditPol/list/category/v

Category/Subcategory	GUID
Access to objects	{6997984A-797A-11D9-BED3-505054503030}
DS access	{6997984F-797A-11D9-BED3-505054503030}
Change of strategy	{6997984D-797A-11D9-BED3-505054503030}
Account logon	{69979850-797A-11D9-BED3-505054503030}
Account management	{6997984E-797A-11D9-BED3-505054503030}
Logon/Logoff	{69979849-797A-11D9-BED3-505054503030}
Detailed tracking	{6997984C-797A-11D9-BED3-505054503030}
System	{69979848-797A-11D9-BED3-505054503030}
Use of privilege	{6997984B-797A-11D9-BED3-505054503030}

To see which security events plotted currently:

```
auditpol.exe/get/category: *
```

The two commands to launch:

```
' auditpol.exe/set/category: {6997984C-797A-11D9-BED3-505054503030} /success:enable success /failure:enable"
```

```
' auditpol.exe/set/category: {69979848-797A-11D9-BED3-505054503030} /success:enable success /failure:enable"
```

The command, which generates a BSOD of Winpe:

```
' auditpol.exe/set/category: {6997984A-797A-11D9-BED3-505054503030} /success:enable success '
```

The firewall: its logs, its configuration, profiles

A few commands used by gatherNetworkInfo.vbs for the current status of the firewall:

```
netsh advfirewall monitor show currentprofile
```

```
netsh advfirewall monitor show firewall
```

```
netsh advfirewall monitor show consec
```

```
netsh advfirewall firewall show rule name=all verbose
```

```
netsh advfirewall consec show rule name=all verbose
```

```
netsh advfirewall monitor show firewall rule name=all verbose
```

```
netsh advfirewall monitor show consec rule name=all
```

The logs

With auditing, event log records the activity of the firewall. Nevertheless, it is possible to get a more specialized log file. To do this, launch "WF.msc". Then, click on 'properties' (a small link under the latest profile), and activate the log.

☛ Attention: the entries in the log file are behind at least 30 seconds on the event. So, patience!

Activation/Disabling

It is better to leave active service and to authorize all flows. Many applications ask the 'Firewall' and do not work if its API do not respond. Same entry point for configuration as above.

For example, if you stop the firewall service with 'net stop MpsSvc', then the 'ping' to WinPe fail... But if it starts with "net start MpsSvc firewall and be allowed the "inbound connections" with "Wf.msc", then the 'ping' to WinPe succeed.

Useful with the WinPe without MMC: 'Wpeutil disablefirewall' or 'Wpeutil enablefirewall'.

How to change the profile of the public in private network? NOT OK !

Information found on the internet by implementing "NetProfM» (Network List Service) service and DCOM interface:

```
# Get network connections

$networkListManager =
[Activator]::CreateInstance([Type]::GetTypeFromCLSID([Guid]"{DCB00C01-570F-4A9B-8D69-
199FDBA5723B}"))

$connections = $networkListManager.GetNetworkConnections)

# Set network location to Private for all networks

$connections | % {$_. GetNetwork(). {SetCategory (1)}
```

The Center "Network and sharing" allows to see the change. But WF. MSC indicates that the profile is always public: this change is therefore inoperative.

Other sites

With GUI 'home folder '.

<https://tinkertry.com/how-to-change-Windows-10-network-type-from-public-to-private#Fix2WiFi>

With the registry

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\NetworkList\Profiles

then in the subkeys, the 'Category' describes the profile type value:

Public: (leave this blank) Private: 1 Domain: 2

With Powershell

<http://www.tenforums.com/tutorials/6815-network-location-set-private-public-Windows-10-a.html?LTR=N>

Get-NetConnectionProfile

Set-NetConnectionProfile - Name "xxxxxxxxxxx" - NetworkCategory Private

Information on NLA and Firewall:

<https://blogs.technet.microsoft.com/Networking/2010/09/08/network-location-awareness-NLA-and-how-it-relates-to-Windows-Firewall-profiles/>

The Center "network and sharing : with build 14393 ?

You must install the service «NetProfM. And to make this operational, it is necessary to temporarily neutralize the "SystemSetupInProgress" key during the startup of this service.

Not really useful!

WinRm in the adm session: OK in both directions

The WinRm service (previously named WsMan in windows 8) can send PS commands to a remote

machine. It also allows to receive.

It uses two ports, one (47001) to configure, the other (5985) to receive commands from a remote computer.

After it starts, we can check that it has opened the TCP 47001 port with the command

Netstat to get the pid (or name with - anb) and a consultation if necessary loop!

```
NetStat -ano
TCP 0.0.0.0: 47001    0.0.0.0:0    LISTENING    4
TCP 0.0.0.0:5985    0.0.0.0:0    LISTENING    4
```

WinRm requires a step of configuration with the command "Winrm quickconfig q. Then non-domain, it is necessary to declare trust machines.

```
Winrm Quickconfig -q
Winrm Set winrm/config/client '@{TrustedHosts="*"}
```

But in the State, these commands fail.

Deficiencies: various messages 'access denied '.

What to do? change the 'LocalSystem' WinRm service account or change the content of local groups as well:

```
net localgroup WinRMRemoteWMIUsers__ /add
net localgroup administrateurs system /add
net localgroup administrateurs localservice /add
net localgroup administrateurs networkservice /add
```

Is it an account 'winrm' in the ACL?

I still got the same error 'access denied '. And then a stroke of luck which depends on Serendipity (a question of the game of thousand euros in france).

Discovered by chance, with the start of the "File server" service, we advance a little and it gets the error already encountered in winpe4 or 5 for Windows 8.1.

```
Net start lanmanserver
```

I also change the password for the account "Administrator" to establish "inbound" remote connections. I share a resource for any copies of files. And I install service "SecLogon.

```
net user administrateur +Noel
net share monx=x:\
```

The new message for "Winrm QuickConfig q" becomes:

```
WSManFault
  Message
    ProviderFault
      WSMANFault
        Message = WinRM firewall exception will not work since one of the network connection
types on this machine is set to Public. Change the network connection type to either Domain or
Private and try again.
```

This is a warning indicating that the TCP 5985 port is not opened for traffic entering WinRm.

Configuring trusted machines succeeds.

```
Winrm Set winrm/config/client '@{TrustedHosts="*"}'
```

- For outbound orders WinRm

PowerShell is a control of WinPE and indicates that the remote access with WinRm does not work in Winpe. You can temporarily neutralize the MiniNt key and launch Powershell. And thus the outgoing WinRm commands succeed.

To temporarily neutralize the MiniNt key :

```
reg DELETE HKLM\system\currentcontrolset\control\miniNt /f
start-process PowerShell -argumentList '-executionpolicy unrestricted'
start-sleep -s 1
reg ADD HKLM\system\currentcontrolset\control\miniNt /f
```

To verify that WinRm outbound commands work:

```
$s="192.168.0.12" # my win10 computer
# ask the password for the remote computer
$c=get-credential WIN-LBH1HBGLMAA\noel
invoke-command -computername $s -credential $c -scriptblock {$env:computername}
invoke-command -computername $s -credential $c -scriptblock {gwmi win32_bios}
```

- For incoming orders WinRm

I open the port with netsh.

```
netsh advfirewall firewall Add rule name="NONO-5985-winrm" dir=in protocol=tcp localport=5985 action=allow
```

To open the port, must be that the MpsSvc Firewall service is started. It seems that "wpeinit.exe" who start it.

And with the WMD script session launches more this command at the right time. I modify my PS scripts to open this port.

For testing, I manually run the following command from a console:

```
net stop mpssvc  
WinRM quickconfig  
net start mpssvc
```

And so the port 5985 is well «listening» in "netstat - ano".

The "test-wsman < ip of winpe >" succeeds when it set launched from a remote pc services10.

Question: should it really switch from "public" to "private"?

<http://www.tenforums.com/tutorials/6815-network-location-set-private-public-Windows-10-a.html?LTR=N>

From a remote pc, I run the following commands to the machine «winpe»:

```
$s="192.168.0.15" # my winpe computer  
# ask the password for the winpe computer  
$c=get-credential MINWINPC\administrateur  
invoke-command -computername $s -credential $c -scriptblock {$env:computername}
```

I get the message "[192.168.0.15] the connection to the remote server 192.168.0.15 failed with the following error message: the WSMan service was unable to launch a host process to process the given request. " Make sure the provider host and proxy WSMan are printed correctly. »

Trace Procmon shows that the Winrm service does not launch the program "wsmprovhost.exe".

I modify the following key in the Winpe machine:

```
hklm\system\setup\SystemSetupInProgress = 0
```

Then I raise the "invoke-command" from the machine services10.

And the command is successful!

BITS: operational only with the Adm session

The BITS service requires 'assemblies' and very few files (qmgr.dll mainly).

It is operational in the session administrator but not in the session "system".

The command "Import-Module BitsTransfer" is not necessary in my context.

- The command succeeds in the adm session
- The anomaly with the session «system»:

```
net start bits  
Start-BitsTransfer "http://noel.blanc.free.fr/index.php" "x:"
```

At the launch of ' Start-BitsTransfer, we get the following error:

Start-BitsTransfer: the requested operation was not performed because the user is not connected to the network. The specified service does not exist. (Exception from HRESULT: 0x800704DD)

Ditto with bitsadmin.exe:

```
BITSAdmin.exe /TRANSFER "toto" prerequisite "http://noel.blanc.free.fr/index.php" "x:\"
Unable to add file – 0x800704dd
The requested operation was not performed because the user is not connected to the network
```

.My single track for the moment is to look for information on this error.

- A page of Msdn providing the list of errors:

[https://msdn.microsoft.com/en-us/library/Windows/desktop/ms681381\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/Windows/desktop/ms681381(v=vs.85).aspx)

Usually this happens when the task is configured to run only when the user is logged on.

- Another page on Msdn:

[http://TechNet.Microsoft.com/en-us/library/cc720473\(v=WS.10\).aspx](http://TechNet.Microsoft.com/en-us/library/cc720473(v=WS.10).aspx)

« ERROR_NOT_LOGGED_ON:

The SENS service is not receiving user logon notifications. BITS (version 2.0 and up) depends on logon notifications from Service Control Manager, which in turn depends on the defined service.

Ensure that the service is started and running correctly. »

☞ I install service "SENS" and the service it depends "EventSystem.

But the error is still present in the session "system"!

MSTSC and TermService

- WinPE to my Windows10:

The error message is as follows: "The remote computer Network Level Authentication which your computer does not support requires."

Can be the key "LSA \LmCompatibilityLevel»? " No, with 2, or 3.

- my Windows10 to Winpe: "unable to connect" even after modify the "inbound connections" in the firewall.

The "TermService" service reports that it starts. But in the event log "Windows/TerminalServices-localSesionManager", a message "remote desktop services do not logon because the Setup program is running".

The "TermService" service is not listening on port 3389. Fact with "netstat – ano".

MSTSC from WinPe to Windows10 : it's possible

Site :

<https://technet.microsoft.com/en-us/library/cc732713.aspx>

Network Level Authentication is an authentication method that can be used to enhance RD Session Host server security by requiring that the user be authenticated to the RD Session Host server before a session is created.

To determine whether a computer is running a version of Remote Desktop Connection that **supports Network Level Authentication**, start Remote Desktop Connection, click the icon in the upper-left corner of the Remote Desktop Connection dialog box, and then click About. Look for the phrase Network Level Authentication supported in the About Remote Desktop Connection dialog box.

<http://vidmar.net/weblog/archive/2007/05/27/The-remote-computer-requires-Network-Level-Authentication-which-your-computer.aspx>

Pre-verification of the MSTSC on WinPe configuration

- run MSTSC
- left-click on the icon in the upper left corner or right click in the title bar
- "About" menu

A window opens stating if NLA is supported. By default, NLA is not supported in WinPe.

Disabling NLA on the machine services10

There are at least a configuration that works: disable the network level authentication on the remote machine services10.

On machine services10, part of the menu 'properties system\remote', it is necessary to uncheck the « allow only connections from computers running the remote desktop with network level authentication ».

You can connect to the remote machine services10.

Activation of NLA on WinPe

But is it possible to configure authentication NLA on WinPe? YES!

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig

Security Packages Reg_Multi_Sz kerberos, msv1_0 , tspkg, pku2u, cloudAP, wdigest, schannel

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders

SecurityProviders = credssp.dll

new files : tspkg.dll, credssp.dll

Win10pCap and Wireshark out boot.wim



I make the following choice: presented here installation is launched after startup of WinPe. This means that the installation will be to redo every time it starts. It is very fast and it is not needed every day.

But it would be also possible, if files are copied to the correct location on the media, to revive the "installer.exe" program that installs the driver 'win10pcap' in the hive "system". This is what I do in my VM.

We can make the choice to install the win10Pcap driver in boot.wim, either with DISM, or by modifying the hive 'system' (e.g. by modifying the traitement.ps1 script).

➤ My context:

a VM uses a VHD file containing WinPe 'flat' mode.

The installation might be different with an other context, for example WinPe contained in a usb key

➤ Tested versions:

win10pcap 10.2.0.5002 version read inf file

Wireshark 2.0.2

➤ Sites:

www.Wireshark.org

<http://www.win10pcap.org/>

Win10pCap

The installation quickly displays an error message "installation directory must be on a local hard drive.

Two methods to work around:

- edit the MSI with ORCA
- or extract the MSI file with 7z files (sys, inf, cat, etc).

Get ORCA

I found it in the SDK of windows7 (64-bit) which I had kept an ISO.

In the «E:\Setup\WinSDKTools_amd64\cab1.cab» file there are various tools including orca, signtools, makecert...

Opens this file "cab1.cab" with 7z. And extracted the file

« WinSDK_Orca_Msi_5E20C107_DAA3_4D49_AFAE_7FB2594F0CDC_amd64 ».

Adding the extension ".msi" at the end of this name. We check with 7z it contains many an 'msi structure '. You can now install this product « orca.msi. ».

Method 1: Modification and installation of win10pCap.Msi with Orca

Create a backup copy of the file « win10pCap.msi ».

We start Orca and open the file « win10pCap.msi ». In the left part «Tables», select «ControlEvent. On the right side, select line «installDirDlg», the «Argument» column contains "VerifyReadyDlg". We double click on the fields in the column "Status" of this line. We can now change the value. It replaces all the content « WIXUI_DONTVALIDATE... _VALID = '1' » by '1 '. Valid. Save.

Then it installs this Msi file in WinPe (double-click, or open, or...)

Method 2: Extracting and installing the win10pCap driver

Open the file "win10pCap.msi" with 7Z.

Search inside the file « win10pCap.cab ». Open this file. It contains versions for various OS (windows 7/8 and 10, in version 32 and 64-bit). Extract all files in a directory.

And depending on the configuration that you want, copy the three files (sys, inf, cat) in a directory, as well as the files wpcap.dll, paquet.dll-installer.exe.

In Winpe, you need to recreates the following tree:

- « x:\program Files (x 86) \win10pcap\x64\. »
 - wpcap.dll, paquet.dll-installer.exe
- « x:\program Files (x 86) \win10pcap\x64\drivers/win10 »
 - win10pcap.inf, win10pcap.sys, win10pcap.cat

Go in the directory « x:\program Files (x 86) \win10pcap\ ».

Run "installer.exe" : it installs driver and copy dll in system32.

Wireshark

Installation is simple: run the program previously downloaded from the official site.

Do not validate the WinpCap installation proposed by WireShark.

Will be added following two .dll files in the Directory x:\windows\system32:

msvcp120.dll and msucr120.dll

These two files are not in the original provision of windows10Ent Build 10586. I take them on my machine under windows 10 hosting my VM.

What you need to know about installing driver

To install a driver, you must generally need 3 files.

- a .sys file: this is the driver code
- an .inf file: it describes the actions to perform during the installation
- a .cat file: this is the signature of 2 previous files

All drivers are signed in the version 64-bit (quid in 32 bits). This signature is contained in a .cat file. And a .cat file may contain the signatures of several drivers.

Some drivers like HTTP. SYS have no .inf file (although provided by MS).

An inf file can reference another file inf (loading of another driver, inclusion of commands...).

Principle of installing a driver with ".inf" file

The installation of a driver takes place in two stages:

- Preload:

Context: Winpe is inactive, the boot.wim file is mounted in a working directory

The pre-loading of the pilot is to copy the files in the "store" WinPe (Directory «DriversStore» and «CatRoot») and modify various hives including "Drivers."

The signature is verified from the preload.

Two methods :

With DISM	<p>Isolated files necessary, inf, sys, etc. It identifies the necessary '.cat' files with 'signtool.exe'. Mounting the boot.wim file into a directory DISM.exe/Mount-Wim /WimFile:%ImageWimpe% index:1 /MountDir:%Mount% Installing the driver from the source, containing useful files Dism /image:%Mount%/Add-Driver /Driver:C:\winpe10\hdaudio.inf Dism verifies the signature of files Dism is updating the directory 'DriverStore' and the hive "Drivers." We copy the necessary '.cat' files If necessary, we copy the associated files and modifying the hives</p>
Winbuilder	<p>It resumes "manually" Dism.exe activity. I guess it is to isolate the useful elements in the hives 'Drivers', 'System', etc, and the directories such as "driversStore"...</p>

- installation:

Context: active Winpe

After Winpe startup, loading can be:

- automatic if the pilot manages a PNP device
- or manual as for hdaudio.sys. In this case, can use 'DrvLoad.exe '.

Note: loading hdaudio.sys installs the driver automatically hdaudiobus.

The installation is performed by the OS WinPe. The files are copied from the store to the directory of the drivers (...system32\drivers\).

Principle of installing a driver with no .cat file

It happens that some drivers 'inbox' lack of reference to a .cat file in the .inf file. This is the case for PDF and XPS printer drivers. The only method I've found is to use DISM under active OS windows

10. Dism realizes the pre-installation (import in the driverstore). Installing in an active Winpe is to deal with on a case by case basis (drvload ?).

Principle of installing a driver with no .inf file

You have to build the key in "... \system\controlset001\services\...". ». And do not forget the .cat file. See the script and installation of Http.sys, NativeWifiP, Vwififlt.

How to find the pilots .cat file ?

Read in a script of Winbuilder de Win10PeSe:

```
\\The cat file can be found by using the signtool.exe from the Windows SDK 8.0, use:
"signtool verify /a c:\windows\system32\drivers\monitor.sys /v /kp".
```

MS provides program signtool.exe in the SDK. But I do not know how it came on my PC. With a Visual Studio can be...

For example, to find the hdaudio.sys .cat file in the active OS :

```
"c:\Program Files (x 86) \Microsoft SDKs\Windows\v7.1A\Bin\signtool.exe" verify /a /v /kp
c:\windows\system32\DriverStore\FileRepository\hdaudio.inf_amd64_dab2294dc8af0030\HdAudio.sys

Verifying:
c:\windows\system32\DriverStore\FileRepository\hdaudio.inf_amd64_dab2294dc8af0030\HdAudio.sys

File is signed in catalog: C:\WINDOWS\system32\CatRoot\{F750E6C3-38EE-11D1-85E5-
00C04FC295EE}\Microsoft-Windows-Client-Drivers-drivers-
Package~31bf3856ad364e35~amd64~~10.0.10240.16384.cat

Hash of file (sha1): 4417D4DE1E79592F6B38315112935CC87DE46C53
```

One can also find the driver .cat file in the reference (iso file of windows 10) mounted in a folder (right click "mount"):



Caution: Should be absolutely the active OS build where we start signtool.exe as the reference.

Example:

```
"c:\Program Files (x 86) \Microsoft SDKs\Windows\v7.1A\Bin\signtool.exe" verify /a /v /kp
C:\z_Win_OS\w10\mount\windows\system32\drivers\vwifibus.sys
```

Consult logs

The consultation of logs is paramount during the investigations. Therefore, locate useful log.

It often launches DISM to build WinPe. And we launch DvrLoad after startup of WinPe.

Dism.exe	DrvLoad.exe
----------	-------------

C :\windows\inf\setup.dev.offline.logDism Dism displays the path of log file	X :\windows\inf\setupapi.dev.log
---	----------------------------------

Display drivers: driverquery

DriverQuery /?

Display only signed drivers: driverquery /si

Details for other drivers: driverquery /v /fo csv

And a better view with PS:

```
driverquery /v /fo csv > c:\temp\qq.txt
```

```
Import-Csv c:\temp\qq.txt | Out-GridView
```

Trace the loading of drivers when you start Winpe

Modify Bcd thus:

```
Bcdedit/store...\boot.bcd/set {default} bootlog Yes
```

The **x:\windows\NtBtLog.txt** file contains the list of drivers loaded and unloaded. But without explanation. It is sometimes a beginning of track.

Installing the hdaudio drivers

Installing the audio drivers from a builder increases Boot.wim' size and therefore the loading time of Winpe. These specific drivers seem not always necessary.

It seems that driver "hdaudio.sys" is sufficient in many cases to get sounds in Winpe. Its installation requires some comment.

- The hdaudio.inf file does not reference a .cat file, so "drvload" cannot load it under Winpe
- The single-copy in the directory «catroot» .cat is not enough for its installation: must also the hive 'drivers' contain a reference to this driver.

A quick test: take the hive 'driver' for install.wim (but it does not contain the 'fbfw' driver to write to X :)

- Can you pre-install with Dism?

Yes with winpe inactive: Dism /image:%Mount%/Add-Driver /Driver:C:\winpe10\hdaudio.inf

Not with active winpe: the "online" mode is not supported by Dism for Winpe!

- Can you then load it with DrvLoad with active winpe? Yes
- Be it a few additional commands after the startup of winpe? Yes.
- Codecs are visible with msinfo.exe

- The MMCSS driver. SYS is it mandatory? no in my context

All 'audio' formats are not recognized. For example, a file in the MP4 format is not read.

Commands additional post-boot

I made the choice of preload hdaudio.sys driver. With another method of installation (close to WinPeSe) the desACL installation is not necessary.

The following sequence is required after the startup of Winpe :

- AudioEndpointBuilder service startup
- loading the driver hdaudio : `drvload d:\sourceDesAjouts\audio\hdaudio\hdaudio.inf`
- modification of the acl
key : 'HKLM:\SOFTWARE\MICROSOFT\Windows\CurrentVersion\MMDevices\Audio\Render'
This key is created after the AudioEndpointBuilder service starts.
- start the AudioSrv service
- COM registration for wmpayer

See the script for details.

WMPLAYER : OK with build 10586, NOK with build 14393

I tested these file types « WAV » and « MP3 ».

Did not take time to amend certain keys, I copied the source Wmplayer 64-bit directory in the directory "program files (X86)" since it is in this directory that the OS will look for the WmPlayer program.

Weird but I do not take the time to look for a better solution.

Note : with build 14393, it is impossible to read files .Wav. But it is possible to read files MP3.

API mciSendString of WinMm.dll

There are several ways to listen to music. This API is easily implemented in a PS script with a bit C#.

I add in boot.wim a few files and one key into « drivers32 » (see script if needd).

Some Sites :

[https://msdn.microsoft.com/en-us/library/windows/desktop/dd757161\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd757161(v=vs.85).aspx)

error : [https://msdn.microsoft.com/en-us/library/aa228215\(v=vs.60\).aspx](https://msdn.microsoft.com/en-us/library/aa228215(v=vs.60).aspx)

<http://pinvoke.net/default.aspx/winmm.mciSendString>

Sites report an anomaly with the "open" command and type "mpegvideo" used for MP3. But after a few tries, the PS script allows to play Wav and MP3 files. And even if the file name contains spaces.

Example in PS:

```

#
# lets read MP3 and a WAV in WINPE
#
param([string] $fileAudio="")

Add-Type -Name mci -Namespace media -MemberDefinition @"
    [DllImport("winmm.dll", EntryPoint = "mciSendStringW", CharSet = CharSet.Unicode,
SetLastError = true, ExactSpelling = true)]
    public static extern int mciSendString(string lpstrCommand, string lpstrReturnString, Int32
uReturnLength, IntPtr hwndCallback);
"@

if ( $fileAudio -eq "" -or -not(test-path $fileAudio) ){
    "fichier absent : $fileAudio"
    return
}

switch (([System.io.path]::getextension($fileAudio)).toupper())
{
    ".WAV"      {
                $type="waveaudio"
                break
            }
    ".MP3"      {
                $type="mpegvideo"
                break
            }
    default {
                "extension non traitée"
                return
            }
}

$stringOpen = "open " + ""$fileAudio`"" + " Type $type Alias playsound" ;
$stringOpen=$stringOpen
$mediaError = [media.mci]::mciSendString($stringOpen, "", 0, 0);
"mediaError=$mediaError"
$stringPlay = "play playsound wait"
$stringPlay=$stringPlay
$mediaError = [media.mci]::mciSendString($stringPlay, "", 0, 0)
"mediaError=$mediaError"
$stringClose = "close playsound wait"
$stringPlay=$stringClose
$mediaError = [media.mci]::mciSendString($stringClose, "", 0, 0)
"mediaError=$mediaError"

```

Wifi

My references

<http://pcloudletter.co.uk/2011/12/03/windows-pe-builder-script-for-waik-including-wifi-support/>

<http://www.serverwatch.com/server-tutorials/using-netsh-commands-for-wi-fi-management-in-windows-8.html>

To decrypt:

<http://stackoverflow.com/questions/10765860/decrypt-wep-wlan-profile-key-using-cryptunprotectdata>

<http://superuser.com/questions/133097/netsh-wlan-add-profile-not-importing-encrypted-passphrase>

Other sites:

http://blogs.technet.com/cfs-filessystemfile.ashx/_key/telligent-evolution-components-attachments/01-6127-00-00-03-31-62-58/Windows-7-Deployment-Procedures-in-802-1X-Wired-Networks.pdf

<http://www.msfm.org/board/topic/162453-winpe-40-enable-wireless-support/>

What should it install?

For Wifi, there are several components to install:

- the OEM-supplied drivers for the cards

I chose to make a preloading with DISM

- the drivers used in the 'network' layer
 - NativeWifiP
 - Vwifflt
 - VwifiBus: it is automatically installed by the netwifibus.inf include
- the network layer services
 - MS_NativeWifiP
 - MS_vwifi

I chose to make a preloading with DISM. And they will be installed with netcfg.exe

Identify and collect files inf, sys, cat

- The NetServices for netcfg.exe
 - netwifi.inf for the MS_NativeWifiP service
 - netwifflt.inf for the MS_vWifi service
- My wifi card driver (intel in my case)
 - NetWew00.inf, NetWew00.sys, NetWfw00.dat

My card driver .inf file has a 'include':

See the windows\inf\setupapi.offline.log file

- For the include:
 - NetWifiBus.inf, VwifiBus.sys, VwifiBus.sys.mui (en - us or En-us if beta)

The "dism/add-driver" command will fetch the driver in the \inf directory:

```
%Mount%\windows\inf\vwifibus.sys
```

Therefore copying this file NetWifiBus.inf in this directory.

Copy the L2Schemas directory

This directory is available in the ISO reference.

Profile Wifi : The basic NetSh commands

It is sometimes useful to export its Wifi profiles since its Windows10 and import them into the Winpe. In a Win10 machine, configure Wifi connections. Then, export these configurations in Xml files. It will contain the SSID and passwords.

After WinPe startup, you will need to import the desired profile. Then connect.

- Export of profiles registered with password in clear in the current directory:

```
netsh wlan export profile key = clear
```

- Export a profile:

```
netsh wlan export profile name = "freebox_opfm" folder="%SourceDriversWifi%\Profils" key = clear
```

- Import a profile in Winpe:

```
netsh wlan add profile filename = "x:\MesProfilesWifi\freebox_opfm"
```

- Wifi connection with the profile page imported into Winpe:

```
netsh wlan connect name = "freebox_opfm" ssid = "freebox_opfm"
```

A connection Wifi in PS script

On the CodePlex site <http://managedwifi.codeplex.com/> I found a C# code that allows the Wifi connection. I placed it in a PS script. I added a graphical interface as well as the function of disconnection.

On my PC, i need to disable/enable the wifi card before connecting. Without this, the identification of the network fails. I've no explanation.

Printers in Winpe

Printers are very different from each other (multi-function or not, laser or inkjet, etc). They can be connected in various ways, network, usb, wifi, web... The architecture of the pilots has evolved

versions of the OS (kernel, mode user mode...).

Find the right driver is a sensitive issue. It is easier to let the system find drivers alone. It is what is done with the shared printers on a server. During the installation of the printer, drivers contained in the server are automatically downloaded in the workstation.

In the case of the installation of a USB printer, necessary drivers must be available to Winpe either in shipping them in the "boot.wim" or in another medium, USB for example.

I have only one printer, which limits my investigations. I can share it on the network, or connect it to USB.

- Architecture reference site : <https://technet.microsoft.com/en-us/library/cc976755.aspx>
- to launch the GUI « add printer » :
 « "X:\windows\System32\rundll32.exe" printui.dll,PrintUIEntryDPIAware /il »
- view the log file : « x:\windows\system32\inf\setupapi.dev.log »

Printers not visible in control panel

☛ Printers are visible in notepad (menu « print ») but not visible in the control panel « devices and printers ».

Printers are visible with powershell, Get-printer and gwmi Win32_printer

Peculiarities of my printer

It's the « samsung SCX - 4500 Series » printer. 64-bit drivers nevertheless use 32-bit programs. So WOW64 is required in this case. For now, I'm not interested in the function "scan."

Network printer in the session « administrator »: it works

With DISM, I installed the « InBox components » NTPRINT.INF and USBPRINT.INF. We could add « usbscan » if necessary in the same way.

After you have copied the files and registry keys, the "spooler" service starts.

From the « configuration/devices and printers Control Panel » we can begin installing the network printer. It must provide the "credential" of a user authorized to manage the printer and the print queue. Check the various parameters of sharing the printer on the server.

First problem: the printer is clearly visible in notepad (menu "print"), but she does not appear in Control Panel

Network printer in the session « system »: impossible

Installation normally start. The big .cab file is well downloaded. But the finalization (text appearing in the installation window) will display « cannot connect to the printer: error 0x00000534 ».

Procmon trace analysis and comparison with installation on another PC confirm download of drivers and divergence when LSA consults the SAM. The program "PrintIsolationHost.exe is not launched in the case of the error.

The « Spoolsv.exe » activity at that time:

- Enter the name of the account and the password of the user created on the server
- creation of the registry key « \software\microsoft\windowsNt\CurrentVesion\print\providers\client Side Rendering Print Providers\servers\...\Client Port\ Side... ».
- Download the .cab file containing the drivers
- display «finalizing...» (not always visible)
- deletion of the previously created key
- display of the error message « error 0x00000534 ».

During a search on MSDN for error codes, I found this:

« error 0x00000534: No. mapping between account name and SID was done ».

Why this control? Maybe because a printer is an attribute of one account (a bit like in the AD). Test with "SystemSetupInProgress"?

Impossible for me to go further.

Msdn site describing errors: <https://msdn.microsoft.com/en-us/library/cc231199.aspx>

USB printer in the session « administrator »

By installing this USB printer on another pc, and analyzing the file « ...\\inf\setupdev.log », we find the directory containing the correct drivers in the « driverstore ».

This allows then to attempt installation on Winpe, either with the Control Panel, or from Device Manager by selecting the printer and not the scanner in my case.

The installation is successful. The printer is visible in « notepad/menu print ».

But printing fails with the message « a StartDocPrinter call was not operated ».

The procon trace shows that the spooler (spoolsv.exe) tests the key "SystemSetupInProgress.

The workaround:

- SystemSetupInProgress = 0
- start the spooler service and starting the print at least once, before StartDocPrinter
- SystemSetupInProgress = 1

Printing is operational. But again, the printer is not visible in the « configuration/devices and printers Control Panel ».

It is easier to install the printer from the Device Manager after connected it (avoids creating a port and a printer ghosts).

The USB printer installed in a session is visible and available in the other session.

USB printer in the session « system »

Installation and printing work as in the case of the session « administrator ».

The scan of my « samsung » : required manual processing

At the end of installation of printer and scan, some keys are not filled. Add the missing keys and stop/start the « stisvc » service.

Keys to add :

```
HKLM\SYSTEM\CurrentControlSet\Control\Class\{6bdd1fc6-810f-11d0-bec7-08002be2092f}\0000
"SubClass"="StillImage"
"friendlyname"="My scan Samsung"
"DeviceID"="{6bdd1fc6-810f-11d0-bec7-08002be2092f}\0000"
"capabilities"=dword:00000031
"deviceType"=dword:00000001
"deviceSubType"=dword:00000001
```

☛ The presence of an index (here 0000) complicates the automatation

☞ Architecture of WIA :

<https://msdn.microsoft.com/en-us/windows/hardware/drivers/image/wia-architecture-overview>

<https://msdn.microsoft.com/en-us/windows/hardware/drivers/image/wia-core-components>

☞ to activate the traces of WIA :

The file log : \debug\wiadebug.log.

The modification of keys :

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\StillImage\Trace\wiaservc.dll]
```

```
"TraceFlags"=dword:00000007
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\StillImage\Trace\sti.dll]
```

```
"TraceFlags"=dword:00000007
```

PDF and XPS printers

It's the spooler that fully and automatically installs these two printers. After the spooler service starts, it takes 2 minutes before automatically begins the start of installation.

In Device Manager, PDF and XPS printers are located in the item « printers » and with the name « unknown device ». To get correct, run the command (not mandatory) :

```
DRVLOAD X:\windows\System32\DriverStore\FileRepository\printqueue.inf_amd64_293deb0d10d72f40\printqueue.inf
```

Then the printers are displaced in the item "print queue" with their correct name.

Note: the spooler activity observed with procmon during automatic installation of these two printers

installation of 2 packages

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\PackagesToAdd\
ntprint et prnms003

launch of

X:\windows\system32\regsvr32.exe /s

"X:\windows\system32\spool\drivers\x64\3\PrintConfig.dll"

installation of service PrintNotify

HKLM\SYSTEM\CurrentControlSet\Services\PrintNotify

X:\windows\system32\spool\drivers\x64\3\PrintConfig.dll

search packages :

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\PackagesToAdd\
...

service "DeviceInstall" (umpnpmgr.dll) launches Drvinst.exe

HKLM\SYSTEM\CurrentControlSet\Services\DeviceInstall\Parameters

%SystemRoot%\system32\umpnpmgr.dll

« **DrvInst.exe** "4" "d" "X:\windows\system32\spool\tools\Microsoft Print To PDF\prnms009.Inf" "9"
"4cf18dca3" "0000000000000670" "WinSta0\Default" "00000000000003AC" "208"
"X:\windows\system32\spool\tools\Microsoft Print To PDF" »

call to service Windows Update absent : X:\windows\System32\wuaueng.dll

write the key HKLM\DRIVERS\DriverDatabase\DeviceIds\{4d36e979-e325-11ce-bfc1-
08002be10318}\

work of service HKLM\SYSTEM\CurrentControlSet\Services\CryptSvc

copy cat file

src :X:\windows\system32\DriverStore\FileRepository\prnms009.inf_amd64_bd3f6a64dee15
35d\prnms009.cat

dst : X:\WINDOWS\SYSTEM32\CATROOT\{F750E6C3-38EE-11D1-85E5-
00C04FC295EE}\oem14.cat

control : X:\windows\system32\catroot2\dberr.txt

In the same time, construction big cab file

X:\windows\system32\spool\drivers\x64\PCC\ntprint.inf_amd64_7b3eed059f4c3e41.cab

write key describing the printer pdf

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\Printers\Microsoft Print to
PDF

Construction cab file

X:\windows\system32\spool\drivers\x64\PCC\prnms009.inf_amd64_bd3f6a64dee1535d.cab

Create files

X:\windows\system32\spool\V4Dirs\50C45F04-5F15-4B72-A323-
05EBFFE9D913\device_bidi.gpd

X:\windows\system32\spool\V4Dirs\50C45F04-5F15-4B72-A323-
05EBFFE9D913\merged.gpd

...

read of HKLM\System\CurrentControlSet\Control\Print\EnableJavaScriptDebugging
printconfig.dll calls jscript.dll

work unknow

```
create HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\OfflinePrinters\Fax
delete HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\OfflinePrinters\Microsoft
Print To PDF
```

```
launch of « DrvInst.exe "1" "0" "SWD\PRINTENUM\{024AFB8F-1EF1-4ECF-B5EF-
27F41CC0E988}" "" "" "49d499bbb" "0000000000000000" »
```

```
lecture de
```

```
HKLM\DRIVERS\DriverDatabase\DriverPackages\printqueue.inf_amd64_293dcb0d10d72f40
```

```
launch of « DrvInst.exe "4" "d" "X:\windows\system32\spool\tools\Microsoft XPS Document
Writer\prnms001.Inf" "9" "4ec392ceb" "0000000000000067C" "WinSta0\Default"
"00000000000000678" "208" "X:\windows\system32\spool\tools\Microsoft XPS Document Writer" »
```

```
...
```

```
search of X:\WINDOWS\SYSTEM32\FaxPrinterInstaller.dll
```

```
launch of « DrvInst.exe "1" "0" "SWD\PRINTENUM\{88C0C652-C649-4C6D-8136-
5F8B16982552}" "" "" "455aafd0f" "0000000000000000" »
```

```
read HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Device Installer\DebugInstall
```

WinPE "Flat mode" in a VHD for a VM in HyperV

Please visit the following websites for information on:

- "flat mode" of Winpe: <https://technet.microsoft.com/en-us/library/hh825045.aspx>
- the vhd: [https://msdn.microsoft.com/en-us/library/windows/desktop/dd323654\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd323654(v=vs.85).aspx)
- VHD and native boot: <https://msdn.microsoft.com/fr-fr/windows/hardware/commercialize/manufacture/desktop/boot-to-vhd--native-boot--add-a-virtual-hard-disk-to-the-boot-menu>

You must build a suitable BCD to use Winpe in a VHD. And add the parameter "Winpe Yes". It's simple and yet...

This site deals with the command "bootsect.exe /nt60" :

<https://blogs.technet.microsoft.com/heyscriptingguy/2015/12/04/build-a-powershell-enabled-windows-pe-key-part-5/>

Context used for the commands in this example

The paths are:

Boot.Wim	C:\MicroWinpeBuilder\media\sources\boot. Wim
VHD file	C:\Vhd\Winpe.vhd Note: create the directory if needed
Unit for the VHD	"H:", to mount the VHD (right click or command PS, or Dispart. or disk management..)
BCD to alter	H:\boot\bcd

The five stages of construction



1 - build the VHD disk (or vhdx) and mount it: several tools are possible

- Diskpart with the following commands:

```
create vdisk file="C:\Vhd\Winpe.vhd" maximum = 9000 type=fixed
attach vdisk
create partition primary
assign letter = H
format fs = ntfs quick
Active
exit
```


To use a diskpart script file:

```
copy this text into a file "CreateVhd.txt".  
run the command "diskpart.exe /s CreateVhd.txt "
```

- with PS commands:

See <https://technet.microsoft.com/en-us/library/hh848503.aspx>

```
$vhdPath = "C:\Vhd\Winpe.vhd"  
$vhdSize = 9GB  
$vhdLetter = "H"  
  
New -VHD -Path $vhdPath -Fixed -SizeBytes $vhdSize | Mount VHD -Passthru | Initialize Disk  
-PartitionStyle MBR -Passthru | New-Partition DriveLetter $vhdLetter -UseMaximumSize -IsActive  
-DriveLetter $vhdLetter | Format-Volume -FileSystem NTFS -Confirm: $false -Force
```

 **2 - decompress Boot.wim in the VHD:** at least 2 efficient tools

- with 7Z, version greater than or equal to 9.20

```
Use the GUI to decompress the Boot.wim file in H:\
```

- With Dism

```
Dism.exe Apply-Image -ImageFile:"C:\MicroWinpeBuiler\media\sources\boot.wim" / Index: 1  
/ApplyDir:H:
```

 **3 - build files (legacy and UEFI) start with bootbcd.exe**

```
BCDboot.exe H:\Windows /s H: /f ALL
```

 **4 - modify the BCD with bcdedit.exe**

The bcd was created by BcdBoot command. But this command ignores Winpe. So, add this parameter.

```
Bcdedit.exe /set {default} Winpe YES H:boot\bcd /store
```

Or with Powershell:


```
Bcdedit/set H:boot\bcd /store '{default}' Winpe YES
```

We get the next bcd:

```
PS C:\WINDOWS\system32> Bcdedit /store H:boot\bcd
```

Gestionnaire de démarrage Windows

```
-----
identificateur    {bootmgr}
device           partition=H:
description      Windows Boot Manager
locale           en-us
inherit          {globalsettings}
default          {default}
displayorder     {default}
toolsdisplayorder {memdiag}
timeout          30
```

Chargeur de démarrage Windows

```
-----
identificateur    {default}
device           partition=H:
path             \Windows\system32\winload.exe
description      Windows PreInstallation Environment
locale           en-us
inherit          {bootloadersettings}
allowedinmemorysettings 0x15000075
osdevice         partition=H:
systemroot       \Windows
nx               OptOut
winpe           Yes
```

5 - eject the VHD

- with explorer: right click then "eject".
- with Diskpart:

```
DiskPart
Select vdisk file="C:\Vhd\Winpe.vhd"
Detach vdisk
exit
```

- with PS:

```
Dismount-VHD - Path "C:\Vhd\Winpe.vhd"
```

WinPE "Flat mode" in a VHD for hard drive

As previously. Only the file BCD is different. In addition this file is not located in the same place.

Indeed, since one will start the PC from a hard drive (external or not, usb or not), the BCD is located in the hard disk and not in the VHD. The VHD is a large size file contained in that drive. In addition, the hard drive can have several partitions or not.

Example of my USB hard drive that contains three partitions F, I and J, as well as several Winpe.

Disque 1	popReserved	winpe (I:)	Maison (J:)	
De base 465,76 Go En ligne	1000 Mo NTFS Sain (Actif, Pa)	9,77 Go NTFS Sain (Partition princ)	415,95 Go NTFS Sain (Partition principale)	39,06 Go Non alloué

The file BCD present into my disk USB :

```
C:\WINDOWS\system32>bcdedit /store f:\boot\bcd
```

Gestionnaire de démarrage Windows

```
-----
identificateur    {bootmgr}
description       Windows Boot Manager
locale            fr-fr
inherit           {globalsettings}
default           {default}
displayorder      {default}
                  {734b7449-e4bb-11e1-a114-001e330ca2a8}
                  {00a5bd49-8f29-11e6-9663-c01885b223e6}
toolsdisplayorder {memdiag}
timeout           30
```


Chargeur de démarrage Windows

```
-----
identificateur    {default}
device          vhd=[I:]\vhd-winpe4-NB.vhd
path              \windows\system32\boot\winload.exe
description       vhd-winpe4-NB
locale            fr-fr
inherit           {bootloadersettings}
osdevice       vhd=[I:]\vhd-winpe4-NB.vhd
systemroot        \windows
detecthal         Yes
winpe          Yes
ems               No
```

Chargeur de démarrage Windows

identificateur	{734b7449-e4bb-11e1-a114-001e330ca2a8}
device	ramdisk=[I:]sources\boot.wim,{7619dcc8-fafe-11d9-b411-000476eba25f}
path	\windows\system32\boot\winload.exe
description	winpe10 build 14393 sur partition2
locale	fr-fr
inherit	{bootloadersettings}
osdevice	ramdisk=[I:]sources\boot.wim,{7619dcc8-fafe-11d9-b411-000476eba25f}
systemroot	\windows
detecthal	Yes
winpe	Yes
ems	No
Chargeur de démarrage Windows	

identificateur	{00a5bd49-8f29-11e6-9663-c01885b223e6}
device	vhd=[J:]mesvhd\vhd2016\test.vhd
path	\windows\system32\boot\winload.exe
description	MicroWinpeBuilder 14393
locale	fr-fr
inherit	{bootloadersettings}
osdevice	vhd=[J:]mesvhd\vhd2016\test.vhd
systemroot	\windows
detecthal	Yes
winpe	Yes
debug	Yes
ems	No

 Attention about ramdisk : This is not because you write « **ramdisk=[I:]sources\boot.wim, {7619dcc8-fafe-11d9-b411-000476eba25f}** » that the « ...boot\bcd » file contains information about the registry entry « **{7619dcc8-fafe-11d9-b411-000476eba25f}** ». Make sure you know from where it comes or that entry is present. I think I saw a MS site that explains this point and how to create this entry..

Supplements on winpe

«install. ESD»: 7Z.exe 64 bits exceeds 15 version

I have no information on this format newly used by Ms.

The beta version of 7Z.exe 64 bits can open compressed files « install. ESD ».

The pxe boot of winpe with tftp32 64 bits version of 'Jounin.

It is rather slow if the boot.wim image has a size of 500 MB or more.

Download the «Jounin» TFTP server

<http://tftpd32.jounin.NET/Download/tftpd64.452.zip>

One could also use the DHCP to a MS Server.

Need to recover files in the boot.wim image. Thus mount this image. And copy the files from <mount>\windows\system32\boot\pxe in a directory that will be the root of the TFTP server. Then copy the contents of the USB drive, so the entire tree of your final winpe (boot, sources, etc.) files in the root of the TFTP server.

The TFTP server configuration is simple enough: check server TFTP and DHCP server, populate the file to load "pxeboot.n12", put a possibly fixed ip on the pc hosting the tftp server, fill in the base directory (the one that hosts all previous files), open the log tab.

Think of open "public domain" in the firewall for TFTP, what should be done because the first launch of TFTP64 'jounin' opens these ports (UAC?), and the firewall ought to warn you (depending on your configuration of course).

Wholesale: connect two PCs directly by a cable "rj45" (no need for crossover cable with standard all modern pc gigabyte), modify the "bios" to activate the "pxe boot. When starting the pc, choose "pxe boot. After requesting an ip address to the DHCP server, the map sends a TFTP frame on the network. The TFTP server on your other pc receives and returns the file «pxeboot.n12» (n12 or com depending on whether you want to act or not on F12 along the way but I can't remember what it is). Then the pc running the program. The latter then loads "bootmgr.exe» from the TFTP server pc. It is indeed «bootmgr.exe» and not «bootmrg »from the root of the usb stick! Then "bootmgr.exe' charge to turn the BCD, and then everything else (boot.sdi,...\sources\boot.wim) from the TFTP server. It's long but it works.

The FBWF RamDisk. SYS: ScratchSpace limited to 512 MB

To change this value, you can use DISM. But if attempts:

```
"dism /image:%Mount% Set - ScratchSpace:1024.
```

Value invalid workspace. Select 32, 64, 128, 256 or 512. »

I have not found another way to modify the code of the driver. After disassemble the code, it must sign the file. Do not reverse the steps!

To disassemble the code of fbwf

To disassemble the driver file, I used "PeBrows64 Pro". Visit the website for more detail:

<http://www.smidgeonsoft.ProHosting.com/>

It is free (if I understand English). If I had not read the documentation on the site, I could never see one line of code. If I got there, everyone can do.

Symbols

At first launch, the software indicates that the environment variable "_NT_SYMBOL_PATH" is absent and to use the mode "as Administrator".

When it disassembles infrequently, it quickly loses good reflexes. I had forgotten the need of symbols to make a comprehensible list.

Visit the website for more detail:

[https://msdn.Microsoft.com/en-us/library/Windows/desktop/ms681416\(v=vs.85\).aspx](https://msdn.Microsoft.com/en-us/library/Windows/desktop/ms681416(v=vs.85).aspx)

So, in a console 'CMD' I run:

```
' set _NT_SYMBOL_PATH = srv * c:\localsymbols* http://msdl.microsoft.com/download/symbols.
```

Then I run

```
"C:\Program Files\SmidgeonSoft\PEBrowse64 Professional\PEBrowse64.exe"
```

Groped, and alternating 'click right/menu', you will eventually find "driverEntry. Then you can see the call to 'FbwfCheckForWinPEBoot'. This function consults the value 'WinPECacheThreshold'. It carries out tests : CMP EAX, 0 x 400 where 0x400 is 1000d and followed by a 'JBE', we can say that this is the limit to 512 MB.

With my PS script, after modify BCD for my USB key to add "TESTSIGNING" and have change the parameter 'WinPECacheThreshold' in the 'system' hive, I make the changes to the program with a "CMP EAX, 0xFF00, injection of the new checksum and I sign the file, I fill in boot.wim: failure!

There may be a change in the value read to account for something like the memory size. So I reread the code and dig a little more.

In the function, the value read is stored in a variable in memory. PeBrosver64 right click to find an option that provides the "cross reference", ie the location of the file where this memory box is used. I quickly find the 'FbwfInstanceSetup' function with the text « FbwfInstanceSetup: Failed to adjust scratch space; status = 0 x %x ». I dig a little. A call to 'ZwQuerySystemInformation', a calculation, a new comparison with 'CMP RAX, 0x40000000' followed by a 'JBE' to the text of the error...

I change the "JBE" in JMP to neutralize the jump to the error code...

And it works, the size of X: is 2 GB in the Winpe.

The signature for test

The more difficult it is to find the 'magic' 3 programs: Makecert.exe, CertMgr.exe, SignTool.exe

I do not know how they got on my PC. With Visual Studio Community can be.

With services10 64bits, each driver must be signed with a test certificate either a 'real' certificate.

A driver developer has 2 possibilities for its (repetitive) tests:

- block verification of the signature on the test PC

MS prevents the installation of such a driver not signed anywhere and by anyone. The developer must act physically on the test PC to block the certificate of a driver checking with:

- either pressing F8 when starting and selecting the option '?'
- either by connecting a PC of debugging... loading the driver requires that the developer based on 'g' in his PC

- use a test deployed on several PC test certificate

For that, you need modify BCD with the option «TESTSIGNING. A 'test Mode' information

will appear at the bottom to right on the screen. Thus the user is warned of the danger.

Example: Bcdedit.exe/store?: \ Boot\BCD - set {default} TESTSIGNING ON

For more information on «The TESTSIGNING Boot Configuration Option»:

<https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff553484%28V=vs.85%29.aspx?f=255&MSPPErr=-2147217396>

Note:

Before setting BCDEdit options you might need to disable or suspend BitLocker and Secure Boot on the computer.

Starting with Windows 7, Windows displays this watermark only in the lower left hand corner of the desktop.

To sign the driver

It must create a test certificate, install possibly on the developer's machine...

The creation of a test certificate with Makecert.exe

For more information:

[https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff548693\(v=vs.85\).aspx](https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff548693(v=vs.85).aspx)

MakeCert-r - pe - ss TestCertStoreName-n 'CN = CertName' CertFileName.cer

How is it happened on my pc:

```
"C:\Program Files (x 86) \Microsoft SDKs\Windows\v7.1A\Bin\x64\MakeCert.exe"?  
CD "C:\Program Files (x 86) \Microsoft SDKs\Windows\v7.1A\Bin\x64\  
MakeCert.exe-r - pe - ss NoelBlancStore-n "CN = CertificatTestingNoelBlanc" CertificatTestingNoelBlanc.cer
```

Installation of the certificate in the developer pc

For more information:

[https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff553506\(v=vs.85\).aspx](https://msdn.Microsoft.com/en-us/library/Windows/hardware/ff553506(v=vs.85).aspx)

Before you sign a file, you must install the certificate on the developer's machine. If the certificate has been generated on the same machine, it is useless.

You should remember that:

- to sign a driver: the certificate can be installed in any store
- but to verify the signature of a pilot signed with this certificate, the certificate must be installed in: "Trusted Root Certification Authorities"

The name of the 'Trusted Root Certification Authorities certificate' store is root.

Where my orders being admin:

```
CD "C:\Program Files (x 86) \Microsoft SDKs\Windows\v7.1A\Bin\x64"  
CertMgr.exe /add CertificatTestingNoelBlanc.cer /s /r localMachine root
```

OK, I see this certificate with certmgr.msc in my pc 'dev'

But how to sign the driver with this certificate?

Test-Signing a Driver File :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff553467\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff553467(v=vs.85).aspx)

Test-Signing Driver Packages :

<https://msdn.microsoft.com/en-us/library/windows/hardware/ff553480%28v=vs.85%29.aspx?f=255&MSPPErr=-2147217396>

The following command shows how to use SignTool to test-sign a driver file.

This example embeds a signature in Toaster.sys, which is in the amd64 subdirectory under the directory in which the command is run.

The test certificate is named "contoso.com(test)" and it is installed in the certificate store named "PrivateCertStore."

```
SignTool sign /v /s PrivateCertStore /n contoso.com(test) /t  
http://timestamp.verisign.com/scripts/timestamp.dll amd64\toaster.sys
```

So my command :

```
SignTool sign /v /s NoelBlancStore /n CertificatTestingNoelBlanc /t  
http://timestamp.verisign.com/scripts/timestamp.dll fbwf.sys
```

Installing the certificate on the test machine (non-winpe)

This procedure is to be used in the case of a non-Winpe windows.

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff547618\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff547618(v=vs.85).aspx)

The test certificates that are used to embed signatures in driver files and to sign a driver package's catalog file must be added to :

- the Trusted Root Certification Authorities certificate store
- the Trusted Publishers certificate store.

Installing a Test Certificate on a Test Computer :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff553563\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff553563(v=vs.85).aspx)

The following CertMgr command adds the certificate in the certificate file CertificateFileName.cer to

the Trusted Root Certification Authorities certificate store on the test computer:

```
CertMgr.exe /add CertificateFileName.cer /s /r localMachine root
```

The following CertMgr command adds the certificate in the certificate file CertificateFileName.cer to the Trusted Publishers certificate store on the test computer:

```
CertMgr.exe /add CertificateFileName.cer /s /r localMachine trustedpublisher
```

Modified and newly signed driver in Winpe installation

It is the case that interests me.

It is unnecessary to installed the certificate.

It is nevertheless necessary that the driver is signed.

Then it modifies BCD to activate the option "TESTSIGNING"

And do not forget to modify the desired value for the size of the RamDisk:

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\FBWF
```

```
WinPECacheThreshol = dword 00000400 if 1 GB
```

```
WinPECacheThreshol = dword 00000800 if 2 GB
```

A collection of useful keys and complement

```
HKEY_LOCAL_MACHINE\SYSTEM\setup
```

the 'SetupPhase' value corresponds to the phases described in the help of AIK

the "AllowStart" key: document!

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\
```

the MiniNt key indicates that the OS is a Winpe

the SystemStartOptions value contains the values of the BCD as:

```
TESTSIGNING MININT ('winpe' in BCD) BOOTLOG and other unknown values.
```

This value is tested by the 'Fbwf.sys' driver when it is loaded which searches for the string "MiniNT". This driver is also seeking in this value the following strings 'EnabledOnAllSkus' and 'WinpeCacheThreshold'.

"The command ' Dism.../set-ScratchSpace ' change the value:

```
«\system\...\services\fbwf\WinpeCacheThreshold».
```

The "\system\...\services\PartMgr\parameters\" key contains the value "SanPolicy". It allows to hide the local disks of the machine when winpe is loaded. It is used by "WinToGo".

Sites

Vhd : <http://go.microsoft.com/fwlink/?LinkId=205691>

Bcdedit :

<https://msdn.microsoft.com/en-us/windows/hardware/commercialize/manufacture/desktop/bcdedit-command-line-options>

[https://technet.microsoft.com/en-us/library/cc709667\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc709667(v=ws.10).aspx)

[https://msdn.microsoft.com/en-us/library/windows/hardware/dn653287\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/dn653287(v=vs.85).aspx)

Part investigation in Winpe

Basic tools: procmon64.exe, procex64.exe, depends.exe, spy++, windbg

With Winpe 64-bit, you must use the 64-bit versions of software (except if the 32-bit subsystem has been installed).

The 64-bit version of « sysinternals » tools is "integrated" into the file downloaded from Technet but is invisible at first glance. It is activated by the 32-bit version which makes it visible during use.

- How to isolate these 64-bit versions in a simple way?
 - Modify the browser settings to show hidden files.
 - Download Promon/Procex from the Technet site. Unpack it.
 - Copy the Promon.exe program on the desktop.
 - Run Procmon. Click the "stop capture" icon
 - On the desktop, there is a procmon64.exe icon (Show hidden files). Copy it elsewhere with, for example, the name Procmon64.exe.
 - Modify/disable the "hidden file" attribute of this new file "Procmon64.exe".
- Understand the context menus of Procmon:
 - "Stack": provides information about the dll and the thread that triggered the current action. This can guide research.
 - "Property": allows to know the command-line, much useful and full of information.

Direct observation with Procmon

An example to fix ideas: to add MMC.exe, the empty shell which launches the 'snappings '.

- Launch Procmon. Active the capture. Then launch mmc.exe.
- Quickly a message indicating an error.
- Stop the capture.
- Examines the trace of Procmon: incomprehensible !

But you learn quickly.

A classic mistake: too many filters. What is missing is sometimes, but not always, in another process called by the process target. Therefore be careful with filters.

Generally, you have to go towards the end. But as the display also registered information in the trace, we must learn to identify them to get to the right information. There, it is the experience and knowledge!

Finally, adding a dll or a program identified as missing, a COM object key, etc, we arrive to identify all missing files/keys.

When MMC.exe starts without error, you get the list of everything what you should add, keys, files (including assembly).

Then it is necessary to implement these changes to next winpe boot automatically. There still several possibilities:

- inject changes into Boot.wim
- or so to write a Powershell script to inject these changes during the next startup.

☛ The boundaries of the investigation with Procmon: calls to services, pilots, COM objects, to the "named pipe", etc, are not visible.

☛ Other objects almost invisible: the system objects are visible with "technet sysinternals" WinObj.exe Version 32 bits only. See my script « MonSetWow64 .ps1 » for that. It displays some objects.

Symboles with procmon

- dbghelp.dll and symsrv.dll

Put the two files dbghelp.dll and symsrv.dll, in the same directory than procmon.exe. These two dll come from ADK/debugger. With the network, you can consult « stack » and receive more informations.

- local cache

You can get a local cache (here [g:\symcache](#)) if you modify an option in procmon, menu « Configure Symbols/Symbols Path » like this (with no space) :

```
srv*g:\symcache*http://msdl.microsoft.com/download/symbols
```

The symbols will be downloaded from the remote site and stored in the disk cache directory in order to expedite the next views.

See : <https://developer.microsoft.com/en-us/windows/hardware/windows-driver-kit>

- Session System

Procmon displays the message « loading symbol...». But the download does not take place and therefore not display.

It is where the local cache is very useful. We start by making the same consultations with the ADM session and cache fills. Then we start again with the session System: procmon consults the cache previously filled.

Investigation strategies

the logic of the "Ant": the unit elements

For each feature, we search the useful elements one by one. And built large lists of keys and files for each feature.

the logic of "the elephant": of global elements

The principle "who can do more can do less", and having noted for example that the keys of the ClassRoot hive are very useful, can copy whole swathes of registry without asking too many questions,

as long as we understand a bit what you are doing.

pieces of the hive 'Software '

From a well chosen source (an iso downloaded and mounted, a PC running Active Win10) export pieces of hive. For example, CLSID, APPID... or even ClasseRoot in its entirety.

The list becomes less long but you lose a little knowledge of retail.

The first pitfall is the loss of data when exporting because some keys are protected by ACLs.

- Method WinPeSe: above all else, change the ACLs of the hives
- Identify with procmon ACL with an error

It is possible also to make exports from Winpe by loading the hive to analyze file.

Then, it is necessary to change some references so that Winpe continues to operate:

- For "currentuser" if it takes data in the hive of his active PC:

replace c:\users\ <noel> by x:\windows\system32\config\systemprofile

- For all the hives:

replace "c:" in "x:"

and "43,00,2A,00" by "58,00,2A,00" for some binary keys

all of the hive 'Software '

Alternatively, take the 'Software' hive in the file "install. Wim. We then do not forget to:

- delete the "runas" in key '...\classes\appid\ '.
- copy the key "".. .windows Nt\winpe "since the hive of winpe in the destination from "install.wim"hive."

This is possible because the install.wim build 10586 software hive already contains of "X:", which is very weird.

In addition, do not forget that this hive contains also the keys for the 32-bit subsystem.

hive "System"


One may also wonder if we can take the whole of the system hive in install. Wim.

I do not find the link that explains these changes.

But I know that it will take to make a few changes: Edit System\Setup\SetupPhase, Cmdline and disable some drivers.

The SetupPhase value corresponds to the phases described in the documentation of the ADK, Audit, Specialize... It directs the behavior of Winpe, Winlogon software in particular.

Do not forget the role of the ProductPolicy key.

 the System\control...\control\MiniNt key is an indicator, as well as "SystemSetupInProgress», of the activity of a Winpe Os. It is tested for example eventviewer.msc and powershell "remote".

hive "Drivers"

Do the test to check if using this hive you can then avoid the additions of drivers with "dism.

I have never done this test.

files

It could also just copy but it will be a very very large volume. And don't forget files of type ".mui" (related to the available languages). The "assemblies" "Dot Net", the drivers, the "unknown", would be available. In a Vm, this may be considered.

But I've never done this test.

comparisons

For simple functionality, it is sometimes useful to compare trace taken on an active services10 and Winpe. For example, on an active services10, with Task Manager, kill explorer. Activates "Procmon. And launch "Explorer.exe" from Task Manager. Do the same with Winpe and compare. This can help sometimes.

a tutorial attempt to add the native desktop

Open my script 'treatment '. Locate function Etape3(). Read the sequence of actions to follow. Carry them out manually. The lists of objects to process are included in the script.

Which can be summarised as follows:

- Prepare the context: mount boot.wim and install.wim
- Load 6 hives source and destination (software, system, default)
- Export parts of registry: the variables \$ClesDeBaseSoftware and \$ClesDeBaseSystem contain the keys to export, modify, import.
- Modify the Acl of the key "OLE".
- Delete the APPID runas (change the "APPID" keys)
- Add a few key (see function AjoutDeCles)
- Inject ProductPolicy
- Copy files: the \$FileToCopy variable contains the list of files to copy
- Unload the hives
- Load desired drivers

Give the exhaustive list of commands to run or the manipulation with "regedit.exe" and "reg.exe"

quickly became a gigantic work.

A tool seems inevitable if we want to reproduce the creation of your WinPe.

Part MicroWinpeBuilder scripts

Construction scripts

Preamble


- The language of the WinPe built: this will be the language of the ISO image.
- ADK, ISO and host: identical version. I have not tested in another configuration.
- I'm not sure that a « install.esd » file is appropriate.

The boot.wim file is generated by the ADK of services10

Existing scripts use the ADK for WinPe tree. It is therefore in the « Media » directory that can be found the file boot.wim.

The script launches only once the construction of the « boot.wim » using « cotype.cmd ».

This file contains all of the packages offered by Ms. Once built by the ADK, it is saved with the name following « boot.wim.AvecPaquetsDeBase.export ». It is this last file that will be copied in boot.wim to be changed by scripts.

 It is possible to take a boot.wim file already created by another tool (e.g. by WinBuilder) and modify it with new scripts (no interest to take existing scripts). This means rename it to « boot.wim.AvecPaquetsDeBase.export ».

The host machine

You must use a machine under Windows10 build 10586. Various programs, such as DISM when adding drivers, make checks on the « .cat » file which may not work with another OS.

The two sources of reference: ADK and Win10 Entr build 10586

ADK

You must download and install the ADK for windows 10 build 10586. The scripts support the launch of «cotype.cmd.

ISO services10 build 10586

The ISO image of Windows 10 build 10586 contains a file « install.wim ».

Need to unzip this file « install.wim » in a directory for reference.

You can modify the scripts if you want to automate this task.

Summary description

GUI

A first script offers a GUI (HMI) rudimentary to capture the three necessary paths:

- the directory will use "cotype.cmd. It must not exist prior to the launch of "cotype.

- the base directory containing the ADK. It is pre-populated if the ADK is installed correctly.
- the directory containing the uncompressed reference "install.wim".

This GUI offers the registration of these data in a file attached to the script (alternative stream).

Treatments will be visible in the tab "console." A log file will be produced at the end of treatment.

Any errors are not treated!

Treatment

It is the most complex main part. It provides the following steps:

- launch of 'copype.cmd' to build the directory tree
- launch of "dism" to add all packages in the ADK: this is very long!
- addition of all identified changes (mine at the moment, the your soon).

The first two steps are performed only once as long as the file « boot.wim.AvecPaquetsDeBase.export » exists. Subsequent launches of the script use the copy of this file. They realize only the third step.

To start from scratch: remove the WinPe directory.

Tree base used by scripts

...\MicroWinpebuilder

ADK put files here

contains log and iso

...\MicroWinpebuilder\Add_Drivers

contains drivers for my printer usb

...\MicroWinpebuilder\Add_Root

contains procmon, winobj, etc

...\MicroWinpebuilder\MesProfilsWifi

contains files used for connection wifi (differents sites)

...\MicroWinpebuilder\Media

create by ADK

The scripts are designed to modify ...media\sources\Boot.wim

...\MicroWinpebuilder\ISO

copy of ...media to build the file iso

Adding in the tree

- Drivers:

2 solutions, change the code of the script « traitement.ps1 » or « Dism add-driver » after having mount the boot.wim file

- Script in the root:

either by modifying the scripts « GUI » and « traitement » if the file is present in the same directory as the script "GUI", either by placing it in the directory « ...\\MicroWinpebuilder\\Add_Root ».

- Printer

See the script because each case is different

The structure of a component to add

For a component, we still find the same elements:

- files
- registry keys
- ACLs
- treatments deferred when you start winpe (to make quick and easy)

To optimize loads/unloads of registry and avoid conflict with « DISM », I was led to burst the processing of a component. The logic of the script becomes:

- preparation of post-processing files
- loading a hive
- for each component, changes in the registers
- unloading hives
- Dism for adding drivers
- for each component, copies of the files

I agree that changing or adding a component becomes quickly complicated in these conditions. But with a little time...

Two additional files: ProductOptions and English

ProductOptions

It contains data relating to the programs authorized in Windows 10. See previous chapter.

English

It is the translation of the manual.

Script MonSetWow64 for the 32-bit subsystem

The origin

The WinPeSe team offers a program that allows to implement the 32-bit subsystem. Not wanting to

use external program, I took the sources written by a Chinese developer and I converted them into c#. Then I filed them in the script to make them readable.

The principle

Read on the internet:

yamingw found a solution for WoW64 in Win10PE

<http://bbs.wuyou.net/forum.php?mod=viewthread&tid=371490>

<http://winbuilder.cn/forum.php?mod=viewthread&tid=204>

Run setwow64. Ntoskrnl is phase 1 initialization testing if the WinPE is running in memory, it does not create a KnownDlls32 kernel objects. This object is populated with SMSS. When initializing a 32-bit system this object was not found in the path is wrong. This procedure only creates a path, no fill. Source refer to the <https://github.com/Google/SymbolicLink-testing-tools>, the CreateObjectDirectory and NativeSymlink merged. Kernel objects when the application exits disappear, so will both fill in Winmain.

SetWOW64 by yamingw. It works by creating the KnownDlls32 kernel object and linking it against X:\Windows\SysWow64 folder. It ~does what smss.exe should do. Note that it does not allow ThinApp packages to work.

Visualization

From a 64-bit environment, you can run this script as well in WinPe as services10. Without parameters, it allows visualization of the « system objects" ». It is rudimentary and request to be improved. But such, launched in WinPe 64-bit, it allows to identify any missing items.

Note: winobj.exe does not exist in 64 bit version.

Creation

With the « create » parameter, it creates the objects needed to run the 32-bit subsystem.

WifiConnexion login script

I picked up a program found here: <http://managedwifi.codeplex.com> I filed it in PS with a GUI script. It is functionally identical to the « netsh ».

I have added the possibility of disconnection.

☁️ cooling with these APIs is long enough, sometimes 30 seconds.

FwFb.sys change script

Get-checksumPrg

It allows to:

- check the checksum of a programme information comparing
 - the checksum included in the file by the linker
 - and the checksum calule by the OS API
- modify this checksum if necessary (param - modify true)

There are several APIs to control these two checksum. I take that which I have already used: MapFileAndCheckSum of imagehlp.dll

[https://msdn.Microsoft.com/en-us/library/Windows/desktop/ms680355\(v=vs.85\).aspx](https://msdn.Microsoft.com/en-us/library/Windows/desktop/ms680355(v=vs.85).aspx)

The modification of the cheksums will be written in the file of the program. The description of the PE file format remains essential.

Sites:

https://Fr.Wikipedia.org/wiki/Portable_Executable

[https://msdn.Microsoft.com/en-us/library/Windows/desktop/ms680336\(v=vs.85\).aspx](https://msdn.Microsoft.com/en-us/library/Windows/desktop/ms680336(v=vs.85).aspx)

modify-bytes

I quickly wrote this piece of script. I used it twice to edit the fwfb.sys file.

It is not optimized. You to complete it if necessary.

In the state, you should do two changes one after another by setting / removing comments.

```
# first change
#[int[]]$lesOctetsAchercherEnHexa = ( 0x72, 0x07, 0x3d, 0x00, 0x04, 0x00, 0x00, 0x76, 0x02, 0x8b,
0xc3, 0xc1, 0xe0, 0x14, 0x89, 0x05, 0xa8, 0x46, 0x01, 0x00, 0x89, 0x05, 0xb2, 0x46, 0x01, 0x00 )
#[int[]]$lesNouveauxOctetsEnHexa = ( 0x72, 0x07, 0x3d, 0x00, 0xFF, 0x00, 0x00, 0x76, 0x02,
0x8b, 0xc3, 0xc1, 0xe0, 0x14, 0x89, 0x05, 0xa8, 0x46, 0x01, 0x00, 0x89, 0x05, 0xb2, 0x46, 0x01,
0x00 )
```

```
# second change
#[int[]]$lesOctetsAchercherEnHexa = ( 0xBA, 0x95, 0x00, 0x00, 0xC0, 0xEB, 0x22, 0x48, 0x25,
0x00, 0x00, 0x00, 0xC0, 0x48, 0x3D, 0x00, 0x00, 0x00, 0x40, 0x72, 0x11, 0xB8, 0x00, 0x00, 0x00,
0x20, 0x89, 0x05, 0x48, 0x50, 0x01, 0x00, 0x89, 0x05, 0x52, 0x50, 0x01, 0x00 )
#[int[]]$lesNouveauxOctetsEnHexa = ( 0xBA, 0x95, 0x00, 0x00, 0xC0, 0xEB, 0x22, 0x48, 0x25,
0x00, 0x00, 0x00, 0xC0, 0x48, 0x3D, 0x00, 0x00, 0x00, 0x40, 0xEB, 0x11, 0xB8, 0x00, 0x00, 0x00,
0x20, 0x89, 0x05, 0x48, 0x50, 0x01, 0x00, 0x89, 0x05, 0x52, 0x50, 0x01, 0x00 )
```

A byte-by-byte PS research takes time. PS research remains possible for a 100 k file.

The name of the file to modify is frozen in the variable:

```
$filePath = "C:\Users\noe\l\Desktop\informatique\fbwf\modif-en-cours-fbwf.sys"
```

You can adapt it to your needs.