

MicroWinpeBuilder

But :

Apprendre à adapter soi-même un Winpe10 64bits en ajoutant, par exemple, MMC (eventlog, devices, disks, services, firewall, etc), le bureau avec la barre des tâches, Ncpa, Netsh, Wifi, MSI, le serveur de fichiers, Audio, Wow64, session adm, Bits, Winrm, MSTCS avec NLA, imprimante/scan usb ou réseau, Bluetooth, Wpd/Mtp...

Dans ce document, je rassemble toutes les informations utiles que j'ai découvertes moi-même ou que j'ai trouvées sur les sites de références :

<http://theoven.org>

<http://reboot.pro>

<https://msfn.org>

Le site de Slore avec qui je partage beaucoup :

<https://gitee.com/slorelee/wimbuilder2>

Sans les conseils et l'aide précieuse des participants de ces sites, je ne serai jamais parvenu à construire et adapter « mon » Winpe selon mon envie.

Ce document sera , je l'espère, utile aux débutants curieux comme moi. Je n'ai pas la prétention de croire qu'un expert y trouvera un quelconque intérêt.

Limite : J'ai commencé ce document avec la version 1511 build 10586. Les versions de Windows se suivent rapidement. Les modifications introduites par MS perturbent l'équilibre précaire d'un Winpe très personnalisé. Donc, ce qui est vrai pour une version sera peut-être faux pour la suivante. En plus, je peux tout simplement me tromper ou faire de mauvais constats.

N'hésitez pas à me contacter à l'adresse ci-dessous. Vos remarques, quelles qu'elles soient, m'intéressent.

A propos de moi : J'ai quitté l'école très tôt, dès 18ans. Je n'ai jamais étudié l'informatique à l'école. J'ai commencé avec Debug.exe sous DOS2.0. Je ne maîtrise aucun sujet. Je sais que mes connaissances sont imparfaites. J'utilise IDA et WinDbg comme un débutant. Je suis un éternel débutant et je sais rester humble.

Présentation succincte du but poursuivi

- Comprendre comment adapter Winpe10 généré par l'ADK
 - Il faut un point de départ et des connaissances de base, souvent fournis par des tiers. Diverses personnes au fil des ans ont abordé ce sujet. CF « mes références ».
 - Il faut aussi faire des choix. Modifier un produit fini comme Winpe de ADK implique que l'on a un besoin, une raison, un but. Certains feront le choix de réduire la taille du fichier Boot.Wim. D'autres, comme moi, feront le choix inverse, ajouter le plus de fonctionnalités possibles.
 - Comprendre, c'est aussi faire des essais parfois infructueux. Dans ce contexte, l'efficacité et la rapidité ne sont pas les critères principaux.
- Apprendre à réaliser les investigations de base
 - La première adaptation consiste souvent à ajouter « le bureau de Explorer ». En cherchant dans diverses sources, on trouve tous les éléments à réunir et à injecter dans le fichier Boot.Wim. Mais lorsque l'on tente d'ajouter une fonctionnalité indisponible dans les sites de référence, on est souvent obligé à mettre en œuvre une investigation nécessitant un débbugger, un désassembleur, des outils comme Procmon...De plus il peut s'avérer utile de faire des comparaisons de ruches avec d'autres sources (comme avec Winpe FullFlat).
- Intégrer les informations découvertes ou mises à disposition sur internet
 - Lorsque l'on dispose des données utiles pour personnaliser son Winpe, il faut les injecter dans le fichier Boot.Wim. La longueur des listes de fichiers et de clés, le caractère optionnel ou non que l'on souhaite donner à certains éléments, tout cela rend incontournable soit l'utilisation d'un produit tiers (que je désigne ici sous le nom de « Winpe Builder »), soit la création de son propre « Winpe Builder ». Pour ma part, souhaitant rester autonome, j'ai écrit le mien en Powershell.
- Apporter un complément d'information pour le débutant.
 - La collecte des données de base est fastidieuse. Souvent, l'information cherchée se cache au fond d'un script d'un « Winpe Builder » ou au fond d'un forum.
 - Mon idée est donc de partager ma collecte d'informations.

Mise en garde

Mes scripts proposés ici (ou sur un des sites de référence) permettent de construire un WinPe. Mais en raison de mes choix (de DotNet en particulier), la taille du fichier Boot.Wim est supérieure à 900Mo. Mais peu importe la taille puisque cette construction vise un but pédagogique.

Mes choix

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 «Winpe Builder » tiers)
 - J'utilise uniquement des scripts PowerShell incluant du C# si besoin.
- Règle 2 : utiliser la ruche Software de Install.Wim (par soucis de simplicité)
 - Au fil des versions, je me suis lassé de chercher le plus petit élément à ajouter. J'accepte de ne pas tout connaître et je me satisfais du résultat visible sur l'écran.
- Règle 3 : ne pas proposer d'options, tous les ajouts sont nécessaires (trop d'interdépendances)
- Règle 4 : inclure les listes d'objets dans le script principal (pour limiter le nombre de fichiers)

- Règle 5 : permettre l'ajout de nouvelles fonctionnalités sans chercher à réduire la taille du fichier « Boot.Wim ».
Il faut définir l'usage que l'on fera de son « Winpe ». Pour ma part, je souhaite ajouter le maximum de fonctionnalités. Cela implique que le fichier « Boot.Wim » sera très gros. Cette vision de mon « Winpe » va à l'opposée de la recherche d'une taille minimale. Cela permet donc de faire des choix structurants en privilégiant l'emploi de ruche entière (software, drivers...). La contrepartie est que l'on ne connaîtra jamais le détail.
- Règle 6 : modifier le code binaire des programmes de Winpe pour simplifier les traitements lors du démarrage de Winpe
Avec la 20H2, je modifie plusieurs fichiers pour contourner les tests des clés « MiniNT » et « SysemSetupInProgress », et aussi pour activer certaines fonctionnalités.

Un « builder » écrit en Powershell

J'apprécie beaucoup Powershell. La possibilité d'inclure des lignes de C# apporte une aide efficace en cas de besoin. Mais quelques limites m'ont tout de même obligé d'employer les 2 logiciels suivants : Reg.exe et Robocopy.exe.

Reg :

- il facilite la copie d'arborescence de clés
- la copie d'arborescence de clés est rapide

Robocopy :

- il permet de copier des répertoires dont les noms dépassent la limite de 256 caractères
- mais il ne recopie pas les fichiers plus anciens de la source. Donc les fichiers plus récents dans la destination ne sont pas écrasés.

Pour les ACL :

- j'utilise des commandes Powershell et du C#

Mes scripts

Je les dépose sur l'un des sites de référence pour ne pas surcharger ce document.

Ils sont tellement mal écrits que je donne ici un point d'entrée dans mon script principal pour une recherche rapide:

Ouvrir mon script «Traitement.ps1». Localiser «function Etape3()». Lire la séquence des actions à enchaîner. Les listes des objets à inclure sont incluses dans le script.

Si vous l'ouvrez avec « powershell_ise », tapez dans la fenêtre de commande :

```
$psIse.CurrentFile.Editor.ToggleOutliningExpansion()
```

Cela fermera les blocs « region » et vous permettra de mieux localiser la partie qui vous intéresse.

Trop de détails tuent l'information

Ce document ne reprend pas tous les détails (clés, fichiers, acl) présents dans les scripts PS.

Email : noelblanc.Winpe@free.fr

Classer et hiérarchiser les informations est une activité complexe.
Et je n'ai pas toujours les idées très claires ni très ordonnées.
Lisez les scripts si vous voulez vérifier la présence d'un fichier, d'une clé...

Structure du document

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

- Les clés utiles
- Les écueils connus
- Des détails pour certaines fonctionnalités ajoutées

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

- Comment ajouter un pilote
- Quels outils d'investigation utiliser
- Des informations « historiques » peu utiles (ce document est aussi mon bloc-note)

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

Des annexes ou des archives...peut-être...et en attendant de les jeter à la poubelle.

Préambule

Ce qui est supposé connu ici

- avoir lu la documentation de WinPe disponible sur les sites de MS
- être familier avec les outils disponibles dans l'ADK, DISM, BCDEDIT, BCDBOOT, etc.
- savoir modifier un BCD
 - utiliser la commande « /store »
 - créer un entrée avec la commande « copy »
 - activer l'affichage du menu de boot avec « /set {bootmgr} displaybootmenu yes »
 - ne pas oublier que le BCD pour WinPe ISO contient les clés pour le RamDisk qui ne sont pas présentes dans le BCD du disque dur après une installation normale d'un 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 redémarrage. Les modifications des ruches ne persistent pas.
- avoir lu les scripts CopyPe.cmd et MakeWinpeMedia.cmd de l'ADK

La base d'un didacticiel pour l'ajout du bureau natif

J'ai fait un choix structurant pour construire mon Winpe (C.F. «Mes choix »):

J'utilise la ruche Software de «Install.Wim»

Avec ce choix, les étapes principales de la construction sont::

- Préparer le contexte: monter Boot.Wim et Install.Wim
- Charger les 6 ruches source et destination (software, system, default)
- Modifier de nombreuses ACL (fichiers et registry)
- Supprimer les «Runas» de APPID (modifier les clés «APPID» nécessaires)

- Remplacer les C: par X: dans les ruches utilisées
- Copier les parties de registre souhaitées: mais quelle liste employée?
- Injecter ProductPolicy (elle contient les autorisations/licences des fonctionnalités)
- Ajouter les clés de personnalisation issues de vos recherches: quelle liste employée?
- Démontez les ruches
- Copier les fichiers issus de vos recherches: quelle liste employée?
- Charger avec DISM les pilotes souhaités

Donner la liste exhaustive des fichiers, des clés du registre, des commandes à lancer, des manipulations avec les divers outils utilisés devient vite un travail gigantesque.

Vous retrouverez toutes ces étapes et les listes dans mon script «Traitement.ps1». Tout y est ! Rien n'y est caché.

Un outil me paraît incontournable si on veut créer et personnaliser soi-même son WinPe.
Il en existe plusieurs. A chacun de choisir ou de créer le sien.

Ecueils et anomalies

Les versions de Winpe se succèdent. Les nouvelles modifications apportées au noyau par MS perturbent souvent les « anciens » contournements apportés pour activer des fonctionnalités.

Les écueils connus: liste à compléter

J'ai rencontré ceux-ci lors de mes premiers essais :

- Modifier les Acl de la clé «OLE» pour DragAndDrop
- Supprimer le fichier WallpaperHost.exe s'il est présent
- Supprimer le fichier windows.immersiveshell.serviceprovider.dll s'il est présent (fullflat)
- Ajouter le fichier imageres.dll de win10 car celui de Winpe est plus petit (dépend de la version)
- Ajouter la clé «Environment» dans la ruche Default de Winpe (sinon perturbation du bureau de ADM : à vérifier)

Anomalies corrigées dans mon Winpe 1511(build10586) : pour mémoire !

- Wpeinit se bloque 5 minutes au démarrage de Winpe
La dll «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.

- Déplacement impossible des 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 impossible de raccourci sur le bureau
il manquait les fichiers «appwiz.cpl» et «osbaseln.dll» ainsi que leurs «.mui»
- Absence de fond d'écran: il manquait productOptions\productPolicy et diverses clés
- snippingtools ne fonctionnait pas: il manquait productOptions\productPolicy
- Le clic droit sur le fond d'écran «display, personnalization» ne lance rien: nécessite le sous système 32 bits 🍷* **NO OK from V 1709**
- Mstsc: OK avec NLA,
- Bits : beaucoup de modifications pour le rendre opérationnel
- Lors de l'ouverture de session «adm», un message d'erreur de Logonui.exe apparaît mais n'est pas bloquant: le fichier Imageres.dll de Winpe est plus petit que celui de Windows. Vérifier ce point pour les nouvelles versions.

Anomalies apparues avec la version 1607 build 14393

- WMP lit uniquement le format MP3: corrigé avec le script V17
- BITS : no ok
- Dans le cas d'un VHD, la session Adm ne s'ouvre pas (?)

Anomalies corrigées avec la version 1809

- L'icône «éjection USB» (Eject USB Storage) : enfin une solution mais avec un énorme délai
- MP4 ok in WMP
- BSOD : "software\...\SideBySide" de la référence perturbe le démarrage de "Services.exe". Au hasard, j'ai injecté ces clés depuis la ruche software de ADK et ça marche : pas d'explication !
- Avec un fichier ISO et lors de l'ouverture de la session ADM, le warning « publisher not verified... » ne s'affiche plus. Il manquait le fichier « smartscreenps.dll »

Anomalies corrigées avec la version 20H2

- Installation correcte de mon scanner (... \control\class\{.. incomplète dans Winpe)
- Modification/Vérification de la procédure de modification de FbWf.sys et de sa signature
- L'ouverture de la session ADM était perturbée : dans le cas d'un boot sur l'ISO, le fichier « usrclass.dat » doit exister. Ceci n'est pas nécessaire dans un Winpe Flat en VHD.
- Le service « BITS » = OK
Il nécessite le service « brokerInfrastructure ». Mais ce dernier ne doit pas démarrer trop tôt

sinon les icônes de la barre de tâches sont affichées avec un retard de plusieurs minutes

Anomalies apparues avec la version 20H2

La connexion à un partage réseau avec l'IHM de « explorer » génère parfois une erreur « serveur RPC ». Mais la commande « net use z : \\... » fonctionne correctement.

Anomalies persistantes ou récurrentes

- 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
- Remote powershell et Wsman? Attention aux assemblies !
- Trace réseau avec netsh : ndiscap.sys démarre, le fichier ETL est généré mais pas le fichier CAB

Partie Connaissances de WinPe

Les 3 piliers d'un Winpe personnalisé

Ce sont les bases actuelles de l'intégration du bureau de Explorer dans Winpe.

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

Depuis au moins le build 10586, mes scripts ne procèdent pas à ce remplacement. Mais, avec la version 20H2, cette modification s'avère de nouveau nécessaire.

ProductPolicy

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 actif car je ne sais pas comment faire autrement.

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

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

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

La modification des « runas »

«Explorer» met en œuvre un ensemble d'éléments COM. Le dialogue COM utilise des clients et des serveurs COM. Le dialogue COM met en œuvre 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, plusieurs sites expliquent qu'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)

Les clés importantes:liste à compléter

Cette liste est très limitée pour l'instant car il n'est pas toujours facile de trouver des informations fiables.

Clé "SYSTEM \SETUP"

CmdLine	REG_SZ	Program to launch (Winpeshl, pecmd.exe...)
SetupType	REG_DWORD	0=Do nothing, show login screen 1=Run CMDLine then REBOOT 2=Run CMDLine then show login screen Important : elle est remise à zéro après le démarrage de Winpe ! Elle est primordiale dans FullFlat et aussi dans le mécanisme de sauvegarde des ruches (pseudo-persistence des ruches)
SetupPhase	REG_DWORD	?
SystemSetupInProgress	REG_DWORD	1 = Le comportement de nombreux programmes est modifié. De nombreuses fonctionnalités sont alors bloquées !
SetupShutdownRequired	REG_DWORD	0=Shutdown Poweroff (Hard power off) 1=Shutdown Reboot 2=Shutdown NoReboot (Soft power off)
AllowStart	key	Elle permet de remplacer la commande « net start... ». Le service inscrit sous cette clé sera démarré automatiquement lors du boot

Sources :

<https://social.technet.microsoft.com/Forums/windows/en-US/b942a34d-c4a7-489c-bb01-45dd65fa9b20/setuptype-and-cmdline-at-hkeylocalmachinesystemsetup?forum=itproxpsp>

Clé "MiniNt"

Elle se situe dans HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\MiniNT.

Elle est créée par un des programmes réalisant le démarrage de Winpe.

Elle trouve son origine dans le fichier BCD, avec la valeur « Winpe = YES ». Lorsque le "loader" de WinPe analyse le fichier BCD et détecte cette valeur, il crée la clé MiniNt.

Il est donc impossible de modifier cette valeur pendant le chargement de Winpe

Des valeurs de la clé «Control»

Clé SYSTEM\CurrentControlSet\Control\

PeBootType	REG_SZ	créée par WpeInit.exe, elle identifie le type du média ayant servi à démarrer Winpe - « Flat » : le démarrage a eu lieu à partir d'un fichier VHD (ou disque dur) - « Ramdisk:OpticalDrive » : le démarrage utilise un
------------	--------	--

		Ramdisk (Boot.sdi). Le démarrage a eu lieu depuis un DVD (fichier ISO dans une VM)
SystemStartOptions		Elle contient certaines valeurs du BCD comme : TESTSIGNING MININT ('Winpe' dans le BCD) BOOTLOG et d'autres valeurs inconnues Lors de son chargement, le pilote 'Fbwf.sys' cherche les chaînes MiniNT, EnabledOnAllSkus, WinpeCacheThreshold. Mais je ne sais pas qui renseigne les 2 dernières.

Des clés «System...\Control\Session Manager

BootExecute	Lance un prg mode « native » (autocheck...) Quand ?
BootShell	Manque d'information (...bootim.exe)
SetupExecute	SetupCl.exe : depuis nt4 au moins, change le SID du computer Poqexec.exe ... : réalise les mises à jours après le reboot

Des clés «System...\services\PartMgr»

System...\Services\PartMgr		
https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/Winpe-storage-area-network--san--policy		
Parameters	SanPolicy	4 = masque les disques locaux de la machine quand Winpe est chargé. Elle est aussi utilisée par «WinToGo». Mais on peut monter les disques avec diskmgr.msc

Des clés «Software...\Windows NT\Winpe»

Instdrive	Pas d'information
CustomBackground	WallPaperHost.exe affiche l'image contenue dans ce fichier Mais ce programme n'est pas lancé si Explorer.exe gère le bureau
DisableRemovableStorageInit	Ne pas initialiser les devices 1394 lors de la recherche de unattend.xml, d'où un gain de temps avec wpeInit.exe

Des clés «Software...\Windows»

\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System

https://docs.microsoft.com/en-us/troubleshoot/mem/configmgr/no-mouse-cursor-during-osd-task-sequence		
EnableCursorSuppression	REG_DWORD	0 = Mouse cursor is not suppressed 1 = Mouse cursor is suppressed l'affichage du curseur de souris évite un écran tout noir entre la fin du chargement des pilotes et le lancement de « explorer.exe » (pendant presque 1 minute)

--	--	--

Le fichier unattend.xml de Winpe

<https://docs.microsoft.com/en-us/windows-hardware/customize/desktop/unattend/microsoft-windows-setup-display>

Il permet à WpeInit.exe de recevoir diverses commandes et paramètres.

Modification due aux commandes DSIM : à compléter

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

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 par l'OS pour 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 ». Un même fichier « .cat » peut contenir les signatures de plusieurs pilotes.

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

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

Principe général d'Installation pour Winpe

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

- Pré-chargement :

Contexte habituel: 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 « DriverStore » et « CatRoot ») et à modifier diverses ruches dont « Drivers »

La signature est vérifiée dès le pré-chargement.

Avec Dism:

On identifie les fichiers nécessaires, inf, sys, etc.
 On identifie le fichier « .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 les répertoires « DriverStore », « CatRoot » et la ruche « Drivers »

- Installation :

Contexte : Winpe actif

Après le démarrage de Winpe, le chargement peut être :

- automatique si le pilote gère un périphérique PNP
- manuel comme pour Hdaudio.sys. Dans ce cas, on peut utiliser « DrvLoad.exe ».
- réalisé par un autre program : Wpeinit.exe installe les pilotes nécessaires pour monter le réseau local

note : Le chargement de Hdaudio.sys installe automatiquement le pilote Hdaudiobus.

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

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

C'est le cas le plus habituel.

Le pré-chargeement a été réalisé avec DISM lors de la construction. Un événement particulier (détection PNP, Wpeinit.exe,...) lance la finalisation de l'installation..

Le gestionnaire de périphériques permet l'installation manuelle de tels périphériques.

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

Certains pilotes « inbox » ne possèdent pas de référence à un fichier « .cat » dans leur fichier « .inf ». C'est le cas pour les pilotes d'imprimantes PDF et XPS par exemple.

Mais ces pilotes possèdent une signature. Cette signature fait partie d'un autre fichier qui constitue la base des signatures de l'OS.

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 possède un fichier « .cat » partagé avec d'autres périphériques/composants. Ce fichier « .cat » est souvent déjà inclus dans la fourniture de base de Winpe.

Il faut construire ou recopier les clés dans « ...system\controlset001\services\... ». Voir le script et l'installation de Http.sys, NativeWifiP, Vwififlt.

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.

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
```

Consulter les logs

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

Pour Winpe Inactif et Dism.exe	Pour Winpe Actif
? :\windows\inf\setupapi.dev.offline.log	X :\windows\inf\setupapi.dev.log

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 d'un pilote modifié manuellement

Voir FBWF.SYS

Fonctionnalités

**Je décrirais ici les informations qui ont été importantes lors de ma recherche.
Je ne propose pas ici de liste de fichiers ni de clés à ajouter.
Reportez vous au script pour chercher ces détails.**

Le bureau Explorer.exe

Pour obtenir un bureau avec un certain confort lors de la manipulation des fenêtres, il faut mettre en œuvre le service Themes et le gestionnaire de fenêtres DWM (Display Windows Manager). Les fenêtres s'affichent alors avec des coins carrés et le changement de couleur de la barre de titre signale la fenêtre active.

Pas très beau mais suffisant

- 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 ?

- 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
- 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.
- **Point à vérifier... mais jamais vérifier car maintenant j'utilise la totalité de la ruche software de l'ISO , ce qui évite la recherche clé par clé** ...\\Software\Microsoft\Windows\CurrentVersion\Explorer\UserSignedIn=1 pour éviter par exemple le délai avant l'apparition des icônes de la barre de tâche.

L'affichage du «WallPaper» reste encore un peu obscur pour moi. Voir plus loin !

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

On peut disposer facilement de la seconde sans mettre en œuvre la première.

Cette 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.

La fonction «Affichage du bureau»

Description

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 » qui est 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 en 2016?

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.

«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
```

La fonction «Open Command Here»

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 fonction «Copy as Path»

Il faut appuyer sur la touche «shift» et faire un clic droit en pointant sur le fichier ou le répertoire dans une fenêtre de l'explorateur.

Je ne sais pas comment rendre permanente cette option -> Résolu : Voir la clé dans le script

Les menus contextuels du « desktop »

Ajouter NCPA, DEVMGMT, « device and printer » ... Voir le script.

Le menu contextuel «NEW»

Depuis le build V1809 : j'avais fait une mauvaise analyse.

Depuis 20h2 : je ne fais pas de traitement particulier pour cette fonctionnalité.

Changer le nom de l'icône «Ce PC»

Ref : <https://www.computerperformance.co.uk/registry/hacks-display-computername/>

Modifier la clé:

HKEY_CLASSES_ROOT\CLSID\{20D04FE0-3AEA-1069-A2D8-08002B30309D}

LocalizedString Expand_SZ «%Computername% %Username%»

Ainsi je vois rapidement le nom de l'utilisateur, « System » ou « administrateur » sur l'écran et je sais quelle est la session active.

Function «CopyProgress»

Avec la totalité des clés de Software de Install.Wim, il reste à trouver les bons fichiers. Il manquait le fichier « chartv.dll ».

Autres menus contextuels

Ne disposant pas de la commande « Win + X », j'ai opté pour ajouter les commandes les plus utiles pour moi dans le menu contextuel en cliquant sur le bureau.

WinSxs

Ce répertoire a un rôle fondamental pour les mises à jour de Windows. Il contient absolument tous les fichiers (sauf les drivers) que Windows pourra mettre à jour. Le répertoire « \system32 » ne contient pas de fichiers. Il contient uniquement des pointeurs sur les fichiers de WinSxs. Ce sont des « hardlink ». DISM crée des « hardlink » lorsqu'ils installe un driver ou un composant. Cela est visible dans le fichier de log « Setupapi.dev.offline.log » par exemple.

Certains des fichiers dans WinSxs sont compressés. C'est le cas pour les fichiers des FOD.

« SxsExp project » permet de décompresser de tels fichiers.

Wow64 : le sous-système 32 bits dans Winpe

Certaines versions de Winpe nécessitaient la création de 2 objets «system». Mais maintenant, ils sont créés par Winpe (voir l'archive en fin de document).

Le sous-système32 nécessite aussi d'enrichir les répertoires «WinSXS» et «SysWow64».

MS EDGE depuis 20H2

Dans le cas de l'utilisation de Winpe dans un VHD, on peut utiliser Edge en recopiant dans le VHD le répertoire : « ISO\Program Files (x86)\Microsoft\Edge\Application ».

Sa taille de plus de 300MB reste surprenante et importante.


Le téléchargement est opérationnel avec la session « System » (contrairement à IE64).


Les fichiers de type « PDF » sont lus sans autre ajout.

Etc.

IE Full64 : presque complet (bientôt une archive)

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 rundll312 de suppression des traces de l'activité de l'utilisateur.

 A noter la présence d'un programme de test :

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

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



« The next key displays the last directory used to store a downloadable file from Internet Explorer, which could give a fairly good idea as to where the user stores his/her files. »

HKCU\Software\Microsoft\Internet Explorer\Download Directory

<https://www.forensicfocus.com/downloads/windows-registry-quick-reference.pdf>

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».

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

J'ai fait le test avec la version 1603 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» :

- Le téléchargement : les fichiers sont bien téléchargés dans un répertoire local mais ne sont pas affichés
- Divers menus ne fonctionnent pas

Session «ADM» :

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

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 backedup (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 :)

Mais je n comprends pas ce qu'il faut faire pour aller plus vite.

Changement du nom du «computer»

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

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

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
```

Winpe est conçu pour utiliser un fichier « Unattend.xml » . Voir la doc chez MS.

Puis le nom du «computer» doit être inscrit dans les 3 clés suivantes :

```
...\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 tous actifs !!! Mauvaise surprise !!!

Le principe général lu dans WinPeSe de Theoven.org

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

Pour utiliser la session « ADM » il faut que le PC ait rejoint un workgroup.

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

Il faut absolument arrêter et modifier la configuration «start= demand» 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 s'il n'existe pas déjà.

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.

La modification de LSM.dll

by NyaMisty in github.com/NyaMisty/PE-LSMHooker

<http://theoven.org/index.php?topic=2881.msg34176#msg34176>

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 un «workgroup».

Pour rejoindre un "workgroup", on peut utiliser l'API "JoinDomainOrWorkgroup" 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
$s.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». On peut aussi utiliser un fichier unattend.xml pour renseigner le nom "Computer Name. Mais quid pour le réseau ?

Le répertoire de profil de l'administrateur : à revoir !

La ruche « Default » de Winpe ne contient pas la clé « Environment ». Le programme « Winlogon.exe » lance « Userinit.exe » et lui passe un ensemble de variables d'environnement dont USERNAME.

Clé « Environment » absente : USERNAME = SYSTEM

Clé « Environment » présente : USERNAME = ADMINISTRATOR (selon la langue)

L'impact est important si l'on utilise comme moi un VHD et une session distante pour se connecter avec l'administrateur. Les deux sessions utilisent le même bureau et/ou un nouveau répertoire est créé à chaque connexion de l'administrateur dans le disque USB. Et l'on perd ainsi les derniers ajouts.

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 🚫 **je ne constate plus ce point**

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» et il n'y a pas de ralentissement.


- ntuser.dat ne contient pas "UserSignedIn" : à revoir !

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 dur 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_

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.old» 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-org.exe
- Copier cmd.exe en logonUi.exe
- 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 la ligne de commande de CMD :

```
gwmi win32_process -filter "Name='logonUi.exe'" | select -expand CommandLine
```

- Lancer alors logonUi-org.exe avec les paramètres de la ligne «cmd».

Lors d'échec, logonUi-org.exe 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» : corrigé

Le fichier « smartscreenps.dll » était absent.

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.

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

J'ai cherché plusieurs années, je crois.

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 le 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'outil «Assistant de mise à niveau» que j'ai utilisé pour forcer la mise à jour vers «Windows Creator» installe sur le disque dur le répertoire «c:\Windows10Upgrad ». Dans ce répertoire se trouve le script «EnablerWifiTracing.cmd» qui réalise la mise en œuvre de cette trace pour le réseau WIFI facilement transposable pour ethernet.

L'installation du pilote ndscap.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 «.Ca » 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

Avec la 20H2, je modifie de nombreux fichiers « .dll » pour contourner les tests des clés « MiniNT » et « SysemSetupInProgress »

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.

A revoir !

WinRm dans la session adm : OK dans les deux sens

Description sommaire

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

Email : noelblanc.Winpe@free.fr

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
```

☞ Avant V1809 : pour ouvrir le port, il faut démarrer le service parefeu MpsSvc. 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 «wsmpvhost.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 !

Avec 20H2

Lors des premiers tests avec 20H2, la configuration de WinRm était en échec. Lors des recherches, j'ai modifié la séquence des opérations « post-démarrage ». Maintenant, l'action « rejoindre un workgroup » a lieu avant la configuration de WinRm. Et WinRm fonctionne correctement.

Wuauserv : mise à jour

Il peut servir pour faire des mises à jour automatique de pilotes.

Il utilise BITS.

A COMPLETER

BITS

Il permet de faire des téléchargements facilement.

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

J'ai rencontré des difficultés diverses et différentes au fil des versions de Winpe.

La dernière a été la recopie de fichier « assembly » dont la longueur du chemin dépassée la limite de 256 caractères. Les commandes PS ne traitent pas de tels items.

On peut utiliser « Bitsadmin.exe » ou Powershell.

Avec 20H2

La commande « Import-Module BitsTransfer » était en échec. Il manquait des « assemblies ». La commande « copy-item » de PS ne prend pas en charge les noms de plus de 256 caractères et une « assembly » n'était pas copiée. J'utilise maintenant « robocopy » pour certaines copies.

Le service BITS nécessite le service « BrokerInfrastructure ». Mais si le service « BrokerInfrastructure » démarre avant « Explorer », alors les icônes s'affichent avec un retard de 5 minutes environ. Donc je fais « net start BrokerInfrastructure » dans un script du démarrage lancé par « WpeInit ».

Remote desktop: MSTSC et TermService

Premiers constats :

- Sens « sortant » de Winpe :

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

- Sens « entrant » 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 Windows10 to WinPe : c'est possible

Je ne décris pas la liste des clés, des fichiers, des pilotes et services. Veuillez consulter le script «Traitement.ps1».

MSTSC from WinPe to Windows10 : c'est possible

GUI de logon

Quand on lance MSTSC depuis Winpe, une boîte noire s'affiche et propose de saisir l'identifiant et le mot de passe. Pour retrouver l'interface graphique de connexion, il faut renommer la clé MiniNt.

NLA

<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 NLA de MSTSC sur WinPe

- lancer MSTSC
- clic gauche sur l'icone 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 version 1607 build 14393

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, credsp.dll

Activation de NLA pour WinPe version 1703 build ???

Un peu de changement avec cette version:

- export/import a key

...\ControlSet001\Control\lsa\CredSSP

- create a key/value

key	value	data
...\ControlSet001\Control\SecurityProviders	SecurityProviders	credssp.dll

AllowEncryptionOracle key

See : <https://support.microsoft.com/en-us/help/4093492/credssp-updates-for-cve-2018-0886-march-13-2018>

Samba

Adapter la clé «LSA\LmCompatibilityLevel» à votre besoin

<https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/network-security-lan-manager-authentication-level>

MSRA «remote assistance»

Il suffit d'ajouter quelques fichiers à l'ensemble déjà apportés par MSTSC et RDP.

«L'expert» fonctionne dans les deux sessions, SYSTEM et ADM.

Mais le «novice» doit utiliser la session ADM dans Winpe.

Quickaccess

Cette fonctionnalité demande peu de travail d'investigation. Là aussi, il faut utiliser la session « Administrateur » pour offrir de l'aide.

IC pour HyperV

Là aussi, pas de difficulté majeure, lire le script pour connaître la liste des éléments à utiliser.

Windows Defender Offline : fonctionnalité peu connue

Le site <http://reboot.pro/topic/20497-run-windows-defender-offline-from-Winpe/> explique comment mettre en œuvre cette fonctionnalité. Le point délicat reste la mise à jour des signatures.

A tester dans Winpe :

- consulter le site <https://support.microsoft.com/fr-fr/help/17466/windows-defender-offline-help-protect-my-pc>
- télécharger le fichier 32 ou 64 bits selon votre besoin (par exemple <https://download.microsoft.com/download/1/E/D/1ED15A90-4014-491D-9C96-DEA70B99DD5E/mssstool64.exe>)
- lancer ce programme « mssstool64.exe »
- choisir l'option « créer une iso »
- monter l'iso « C:\ProgramData\Microsoft\Microsoft Standalone System Sweeper Tool\WDO_Media64.iso » dans G: par exemple
- dans la racine de l'iso, récupérer les 2 fichiers
- dans le fichier Boot.Wim, récupérer le répertoire « G:\sources\Boot.Wim\Program Files\Microsoft Security Client\ »

Installation des pilotes hdaudio : à revoir

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 des ACL 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 : 32bits et 64bits indissociables

Mon constat :

Dans la ruche Software, les extensions de fichiers sont associées à WMP32bits

Test à faire : faire pointer une extension de fichier « Wav par exemple » vers WMP64bits

Actuellement il existe deux possibilités pour utiliser WMP:

- Utiliser uniquement WMP 64

copier le répertoire WMP64bits dans "Program Files (x86)"

avantage : gain de place

inconvénient : tous les formats ne sont pas reconnus, MP3 ok

mise en œuvre testée avec v1709 :

Ouvrir WMP64. Puis DragAndDrop du fichier MP3

Anomalie : le double-clic sur un fichier génère une erreur après un délai de 1 à 2 minutes: "Server execution failed"

Le comportement varie selon l'évolution de mes tests et de mes ajouts.

- Utiliser WMP 64 et WMP 32 comme dans windows 10 « normal »

copier le répertoire WMP 32 dans "Program Files (x86)"

copier le répertoire WMP 64 dans "Program Files"

inconvenient : la taille de Boot.Wim augmente.

WAV, MP3, MP4 = OK

☛ Anomalie avec le compte System :

Le double-clic sur un fichier MP3, WAV ou MP4 déposé sur le bureau ouvre WMP mais aucun son (ni image) car WMP cherche le fichier dans l'emplacement suivant :

X:\Windows\SysWOW64\config\systemprofile\Desktop\Ring05.wav

Ancienne méthode abandonnée

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.

Nouvelle méthode: WMP32 et WMP64

les fichiers MP3, WAV et MP4 sont lus correctement.

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 dans WinRe

Comme on peut le lire souvent, Winre peut servir de base à la personnalisation de son « Winpe ».

Le gros avantage, c'est que MS a déjà intégré de nombreuses fonctionnalités à ce « Winre ».

Email : noelblanc.Winpe@free.fr

C'est le cas de WIFI.

Wifi dans Winpe

New (6 Mars 2018) : <http://www.scconfigmgr.com/2018/03/06/build-a-Winpe-with-wireless-support/>

Mes références

<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>

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/>

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.

Avec 20H2, j'utilise une nouvelle méthode pour éviter netcfg.exe. Voir plus loin.

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

- Les NetServices pour Netcfg.exe

- Netwifi.inf pour le service MS_NativeWifiP
- Netwififlt.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.

Nouvelle méthode pour préparer le réseau

Les commandes « netcfg » deviennent trop nombreuses. J'ai tenté de recopier la méthode de WinRE.

En copiant les clés « NetWork » et « WetWorkSetup2 », on peut éviter ces commandes « netcfg ».

Il faut les prendre sur un windows10 ou dans WinRe.

Microphone

Le microphone utilise maintenant le service « CamSvc ».

Bluetooth

La recherche a été un peu longue mais sans grande difficulté.

Le « pairing » : la lenteur de « DevicePairingWizard.exe » m'avait conduit à écrire mon propre outil de « pairing » beaucoup plus rapide.

Les appareils « BHT-LE » ne fonctionnent pas car il n'y a pas de solution pour réaliser le « pairing ».
Il me semble avoir lu qu'il s'agit d'un choix de MS pour Windows.

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 dans devmgmt.msc: corrigé

Les imprimantes sont bien visibles dans notepad (menu «imprimer») mais elles n'apparaissent pas dans le panneau de configuration : je ne sais plus quelle a été la solution !

Elles sont visibles avec powershell, « Get-printer »et « gmi Win32_printer » après ajout des fichiers «mof».

Imprimantes invisibles dans «devices and printers»: corrigé

Les imprimantes deviennent visibles au bout de 2 minutes après le démarrage des services spooler et DSMSVC.. C'est un délai hardcodé dans un programme du spooler.

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.

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 procmon 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».

Imprimante et "bureau à distance"

Je lance Winpe sur une machine éloignée. Et j'utilise "bureau à distance" (mstsc) pour me connecter à ce Winpe. Mes imprimantes locales sont donc ajoutées aux imprimantes locales de Winpe.

Là encore, le rafraîchissement dans "devices and printers" est long et prend entre 3 à 5 minutes. A VERIFIER

Le scanner de ma «sumsung»

Pendant longtemps je n'ai pas cherché à corriger le point suivant. Mais j'ai fini par trouver l'origine du problème.

L'installation du pilote de mon scanner se déroulait a priori correctement. Mais la partie « class installer » n'était pas traitée. La dll « Sti_ci.dll » n'était pas sollicitée car la fourniture de base de Winpe n'inclut pas tous les composants dans la ruche « system ». L'ajout de la clé « ...\\class\{6bdd...\\0000\\installer32 » résout l'anomalie. C'était pourtant simple à trouver.

☞ 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 valeurs par défaut avec la 20H2 sont : 0x407

Pour mémoire uniquement: 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
```

Le service spooler

Le service spooler est la pièce maîtresse de l'architecture des imprimantes. Il faut considérer 2 points essentiels.

- identifier les éléments utiles pour installer le spooler

Cette première étape consiste à démarrer le spooler sans erreur.

Son installation est assez simple et demande peu de travail d'investigation.

- activer le spooler

Comme beaucoup de services (EventLog, TermService...) son démarrage ne suffit pas à le rendre opérationnel. Winpe est un outil pour installer Windows.

De nombreuses fonctionnalités ne sont pas compatibles avec son efficacité, sa rapidité, sa robustesse, etc

Le spooler est inactif si hklm:\system\setup\systemsetupInProgress = 1

- modification de la sécurité (ACL) de ..\spool\printers

Les imprimantes PDF et XPS et le spooler

Quand on a réussi à installer le spooler, qu'il démarre sans erreur, qu'il est actif, il faut alors installer des imprimantes.

Actuellement, c'est le Spooler lui même qui installe les imprimantes PDF et XPS.

Il le fait 2 minutes après son démarrage (comme dans un windows 10 normal)

Mais il faut 10 minutes pour que les imprimantes apparaissent dans « device and printer ».

Une modification des fichiers est nécessaires pour supprimer ces délais.

Note 1 : j'ai ajouté 2 fichiers « .mof » pour Powershell

```
# pour gmwi win32_printer
x:\windows\system32\wbem\mofcomp.exe x:\windows\system32\wbem\win32_printer.mof

# pour get-printer
x:\windows\system32\wbem\mofcomp.exe x:\windows\system32\wbem\
PrintManagementProvider.mof
```

Note 2 : le fichier xpsrchvw.exe est maintenant absent de la fourniture Windows PRO.

Quelques rappels :

Les imprimantes de la session adm peuvent être différentes des imprimantes de la session system.

Les imprimantes redirigées d'une session "bureau à distance" s'ajoutent aux imprimantes locales.

Le rafraîchissement de « devices and Printers" survient 3 à 5 minutes après la connexion (à vérifier).

Dans la session system, on ne peut pas installer une imprimante partagée par un autre micro-ordinateur

Nouveau choix : modification des fichiers

Je modifie plusieurs fichiers pour éliminer les divers délais et les tests de SystemSetupInProgress.

L'éjection des périphériques USB



Fonctionnalité encore aléatoire

Mon contexte de test: boot sur disque USB et vhd contenant Winpe en mode Flat

La zone de notification en bas à droite affiche une icône qui permet l'éjection de périphériques USB connectés. Or, la ligne identifiant le périphérique n'était pas affichée. Ce qui ne permettait pas son éjection.

L'activation actuelle

Le hasard m'a permis de mettre en évidence la séquence suivante:

```
1 - modifier la valeur : hklm:\system\setup -name systemsetupInProgress = 0
2 - net stop dsmsvc ( dans mon "build", ce service dsmsvc est configuré avec "start=demand" )
3 - net start dsmsvc
note : 2 minutes après le démarrage du service DSMSVC, l'entrée "Eject USB Storage" apparaît si un
drive USB était connecté (j'utilise un disque USB de démarrage pour mes tests)
4 - attendre avant de modifier la valeur systemSetupInProgress = 1
```

Ce service Dsmsvc est utilisé pour les imprimantes PDF et XPS. C'est donc en testant mon installation des imprimantes que j'ai constaté l'affichage de la ligne permettant l'éjection. Il est certainement possible de dissocier l'affichage «Eject USB Storage» et les imprimantes.

Anomalies actuelles

- Délai important : semble corrigé

Avec la nouvelle modification, la ligne "Eject Usb Storage" apparaît avec un retard de 2 minutes environ après le démarrage du service DSMSVC

- blocages aléatoires

Mon disque USB de boot est toujours présent. J'insère une clé USB. Elle devient visible dans l'icone "Safety Remove Hardware and Eject Media" en bas à droite.

Depuis l'icone "Eject Usb Storage", si l'on éjecte la clé USB, la fenêtre "explorer" est remise à jour 1 ou 2 secondes plus tard.

Mais l'icone en bas à droite "Safety Remove Hardware and Eject Media" n'affiche rien tant que le périphérique n'a pas été retiré.

Lorsque l'on a retiré le périphérique alors l'icone redevient opérationnel et affiche mon disque USB.

Mais pas toujours !

Je suppose que l'absence de l'animation graphique « Safe to remove the hardware » joue un rôle.

Nécessite parfois «start/stop DSMSVC»

Parfois la clé USB est visible mais pas la ligne «Eject USB Storage».

WPD/MTP and Smartphones

Un peu d'histoire

J'ai cherché longtemps comment aborder le sujet. Je ne connaissais pas l'architecture WDF/WUMF.

Le point de départ était le mécanisme PNP mis en oeuvre lorsque le smartphone est connecté.

Le fichier setupapi.dev.log signalait une erreur :

```
"!!! DVI: Device not started: Device has problem: 0x25 (CM_PROB_FAILED_DRIVER_ENTRY),
problem status: 0xc0000034."
```

Au bout de nombreuses heures de recherche, j'ai trouvé que le driver "wudfRd" ne démarrait pas.

Puis longtemps après, j'ai trouvé qu'il recherchait un port ALPC :

```
"\UMDFCommunicationPorts\ProcessManagement".
```

Il m'a fallu encore du temps pour comprendre que je devais chercher non pas du côté de celui qui voulait utiliser le port mais chercher du côté de celui qui devait le créer.

Et c'est le hasard qui m'a mis sur la piste du programme "services.exe".

Grâce à IDA, j'ai trouvé le test de "MiniNt". Avec Windbg j'ai pu me prouver que c'était le bon point d'entrée de la solution.

Ensuite, installer tous les pilotes et identifier tous les fichiers utiles a été facile.

Mais la mise en oeuvre de la solution n'est pas simple. Elle ne sera pas pérenne.

- Première option : modifier le programme "services.exe" protégé par "PPL security" impossible pour moi
consulter les sites :

<http://www.alex-ionescu.com/?p=97>

<http://www.alex-ionescu.com/?p=116>

<http://www.alex-ionescu.com/?p=146>

<http://2012.ruxconbreakpoint.com/assets/Uploads/bpx/alex-breakpoint2012.pdf>

- Seconde option proposé par slore : utiliser le mécanisme "AppInit_DLLs"
<https://support.microsoft.com/en-us/help/197571/working-with-the-appinit-dlls-registry-value>

The dllentry of the hook (Nothing more):

```
case : attach to the process
      if the process is winlogon then
```

```

        delete the MiniNt key
        loop/wait : svchost process is created // services.exe test MinNt, doesn't find if,
initialize WDF, launch many services
        create again the MiniNt key
    endif
endcase

```

Vous pouvez aussi consulter le site :

<http://reboot.pro/topic/21879-is-it-possible-to-modify-the-services-exe-program-of-Winpe10-v1809/>

Comment j'ai installé WPD/MTP dans Boot.Wim

Je fais ici un résumé du contenu de mon script avec les risque d'erreur que cela comporte.

Il faut disposer de la dll de hook (voir plus loin les sources) et la copier dans «system32» par exemple

- J'utilise la ruche software de Install.Wim
- modifier la ruche Software pour installer cette dll

```

Set-ItemProperty -path $cible_PS -Name "AppInit_DLLs" -Value "X:\windows\system32\hookMiniNt.dll"
Set-ItemProperty -path $cible_PS -Name "LoadAppInit_DLLs" -Value 1
Set-ItemProperty -path $cible_PS -Name "RequireSignedAppInit_DLLs" -Value 0

```

- Installer plusieurs services/drivers et ajouter des «class»

```

Tmp_System\ControlSet001\services\WPDBusEnum
Tmp_System\ControlSet001\services\WpdUpFltr
;for SmartCard Reader (Later)
Tmp_System\ControlSet001\services\WudfPf
Tmp_System\ControlSet001\services\WUDFRd
;for MTP
Tmp_System\ControlSet001\Control\Class\{eec5ad98-8080-425f-922a-dabf3de3f69a}
;for smartCardReader
Tmp_System\ControlSet001\Control\Class\{50dd5230-ba8a-11d1-bf5d-0000f805f530}

```

- Copier des fichiers

```

$filesWPDmTP=@'
;system32

```

```

Windows\System32\WUDF*
Windows\System32\PortableDevice*
Windows\System32\Wpd*
;drivers
Windows\System32\drivers\WUDFPf.sys
Windows\System32\drivers\WUDFRd.sys
Windows\System32\drivers\WpdUpFltr.sys
;cat
Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-Client-Desktop-Required-Package03111~31bf3856ad364e35~amd64~*
Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-Portable-Devices-multimedia-Package~31bf3856ad364e35~amd64~*
Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-WPD-UltimatePortableDeviceFeature-Feature-Package~31bf3856ad364e35~amd64~*
'@

```

- Installer les pilotes avec `dism /add-driver` depuis le DriversStore de la source

```

Dism ... /add-driver ... winusb.inf /forceunsigned
Dism ... /add-driver ... wpdmtp.inf /forceunsigned
Dism ... /add-driver ... wpdfs.inf /forceunsigned
Dism ... /add-driver ... Wpdcomp.inf /forceunsigned
Dism ... /add-driver ... wpdmtphw.inf /forceunsigned
Dism ... /add-driver ... wudfusbccidriver.inf /forceunsigned
Dism ... /add-driver ... UsbccidDriver.inf /forceunsigned
Dism ... /add-driver ... wpdmtphw.inf /forceunsigned
copie des fichiers ".sys" dans "...\drivers"
copie des fichiers "WpdMtp*.dll" dans "...\System32"
copie du fichier "WpdMtpDr.dll" dans "...\System32 \Drivers\UMDF"
copie du fichier "wudfusbccidriver.dll" dans "...\System32 \Drivers\UMDF"
copie du fichier "wpdfs.dll" dans "...\System32"
copie du fichier "winusb.dll l" dans "...\System32"

```

copie du fichier "winusb.sys" dans "...\inf"

copie des fichiers ".inf" dans "...\inf"

L'enfer des versions dans Winpe illustré avec VirtualBox

Peu importe l'utilité d'installer VirtualBox dans Winpe. C'est un exercice comme une autre.

Et surtout c'est une parfaite illustration des incohérences de versions de « dll » ajoutées au système.

La version de VirtualBox 6,1,16-140961 est disponible en même temps que Winpe 20H2.

La base des fichiers de Winpe 20H2 est en réalité la version 20H1 car MS n'a pas publié une nouvelle version de l'ADK à ce moment là.

Virtualbox nécessite le fichier « OpenGL32.dll ». J'ai donc ajouté ce fichier dans Winpe. Je l'ai copié depuis l'ISO de Windows10 20H2.

Le logiciel VirtualBox s'installe correctement. Ensuite, le lancement du programme « VirtualBox.exe » génère l'erreur « Le point d'entrée CheckIsMSIXPackage est introuvable dans OpenGL32.dll ».

Une recherche avec « Depends.exe » montre que le fichier « KernelBase.dll » de Winpe ne contient pas ce point d'entrée. Or, le fichier « KernelBase.dll » de l'ISO 20H2 contient ce point d'entrée.

On pourrait dire ceci : tout est résolu si on prend le fichier « KernelBase.dll » de l'ISO 20H2.

Mais il est possible que cela génère une anomalie pour un autre composant.

Trace ETW et logman

Todo , it seems it needs the key : ...control\wmi

utilité ?

Fichier .TMF

Test des versions de DotNet

Je connais peu dotnet et j'ai cherché un logiciel pour tester les versions DotNet présentes dans Winpe. J'ai retenu celui de MS car puisque j'utilise Winpe de MS alors je peux bien utiliser leur logiciel de test.

J'ai trouvé ceci : <https://blogs.msdn.microsoft.com/astebner/2008/10/13/net-framework-setup-verification-tool-users-guide/>

On peut extraire l'outil de test dans un répertoire avec la commande « netfx_setupverifier.exe /t:c:\temp /c ». Puis utiliser l'outil de test ainsi "Setupverifier2.exe /a".

C'est un logiciel 32 bits mais qui teste les versions 64bits.

Winpe mode Flat dans un VHD

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» pour initialiser un disque :

<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

A titre d'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 Diskpart ou le gestionnaire de disque...)
BCD à modifier	H:\boot\bcd
Bios	Partition Bios et non UEFI (adaptation facile !)

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 1000 Mo NTFS Sain (Actif, Pa <hr/>	winpe (I:) ← 9,77 Go NTFS Sain (Partition princ	Maison (J:) ← 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 pas 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/alphnumeric-object-in-the-dvd-bcd?forum=win10itprosetup>

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

identifiant	{7619dcc8-fafe-11d9-b411-000476eba25f}
ramdisksdidevice	boot
ramdisksdipath	\boot\boot.sdi

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

FbWf et PXE

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

☞ Cette fonctionnalité ne m'intéresse pas vraiment. Le mécanisme de signature introduit la difficulté à contourner. C'est cela qui me paraît utile de tester.

☞ les projets comme WIMBUILDER2/XPE/SE utilisent un ancien « ramdisk » qui provient d'un ancien produit de MS devenu obsolète (« Windows Embedded »). Mais son fichier fbwf.sys continue de fonctionner dans Winpe de W10.

Peut on utiliser DISM pour modifier cette taille maximale de 512MB ? Si on tente :

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

on obtient : « 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 modifier la donnée « checksum PE », signer le fichier et modifier le BCD.

Pour désassembler le fichier du pilote, j'avais utilisé «PeBrows64 Pro». Consulter le site pour plus de détail : <http://www.smidgeonsoft.prohosting.com/>

J'utilise IDA V7 (devenu gratuit) pour désassembler le fichier du pilote,.

Un peu de lecture avec IDA

[Je fais des corrections avec la version 20H2 car il y eu des changements]

Le point d'entrée est « driverEntry ». On 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 du cache en écriture pour X:\ est de 2Go dans le Winpe avec au moins 4GB de RAM.

Modifier le « checksum PE »

Les fichiers de pilotes en 64bit doivent obligatoirement afficher un checksum valide.

La signature pour test

Il faut installer une partie du SDK pour trouver les 3 programmes 'magiques' : Makecert.exe, CertMgr.exe, SignTool.exe. C'est un peu long.

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 faudra aussi 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&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.

Pour signer le driver modifié

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

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

Ma commande actuelle (20h2) :

```
Cd « c:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\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)

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.

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, est il utile de lancer la comande suivante ?

D'où mes commandes, en étant admin :

```
Cd « c:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64 »  
CertMgr.exe /add CertificatTestingNoelBlanc.cer /s /r localMachine root
```

Le certificat est visible 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
```

Avec 20H2 :

<https://docs.microsoft.com/fr-fr/windows-hardware/drivers/install/test-signing-a-driver-file>

Email : noelblanc.Winpe@free.fr

new one : /t http://timestamp.digicert.com

D'où ma commande :

```
Cd « c:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64 »
SignTool sign /v /s NoelBlancStore /n CertificatTestingNoelBlanc /t http://timestamp.digicert.com
fbwf.sys.new
```

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

Il est inutile d'installer le certificat dans Winpe.

On recopie le fichier FbWf.sys dans Winpe.

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
```

Modifier le BCD pour ajouter « TESTSIGNING

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

N'ayant pas besoin de FBWF dans un VHD, je recopie le fichier d'origine lorsque je construit le VHD.

Particularité pour l'environnement UEFI

J'utilise des VM dans HyperV. Pour démarrer Winpe avec un pilote modifié et signé avec un certificat de test, il faut désactiver « Secure Boot » dans les paramètres de la VM. Je n'ai pas fait le test avec un vrai PC et UEFI mais je suppose qu'il faut aussi désactiver cette option.

Note : le paramètre du BCD « NoIntegrityChecks » n'a pas d'effet sur le boot.

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)

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

Cela est trop lent avec une image Boot.Wim de plus de 1GB.

Les informations qui suivent sont là pour aider ma mémoire.

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.

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 parfois intégrée (cela dépend des versions) dans le fichier téléchargé sur Technet mais est invisible de prime abord. Elle est révélée par la version 32 bits qui la rend visible pendant son utilisation.

- 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.
 - Colonne TID = « Thread » ID, pour mieux comprendre le contenu de la pile
 - Ne pas oublier les symboles
 - "Property" : permet de connaître la ligne de commande, bien utile et riche de renseignements.
- Le fichier de configuration : il augmente le confort

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).

☛ 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

La meilleure solution consiste à télécharger le debugger (ADK ou DDK), à placer son répertoire dans la racine de Winpe, et à modifier les options de Procmon pour qu'il pointe sur ce répertoire

Une moins bonne solution consiste à placer les deux dll, dbghelp.dll et symsrv.dll, dans le même répertoire que procmon.exe. Ces deux dll proviennent de l'ADK/debugger.

Ainsi Procmon ira chercher les symboles dans le serveur de MS.

Et avec un réseau opérationnel, la consultation de la « stack » devient plus explicite et donne de meilleures informations.

- 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.

Le cache local peut être 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.

Après de nombreuses recherches, j'ai enfin trouvé l'explication et la bonne solution. Il faut créer la clé :

```
reg add « hklm\software\Microsoft\Symbol Server » -v NoInternetProxy -v 1 -t Dword
```

Capture depuis le démarrage

<https://blogs.msdn.microsoft.com/scw/2015/10/27/how-to-enable-boot-logging-in-windows-10/>

Présentation du mécanisme utilisé :

Le pilote Procmon24.Sys est installé avec le menu de promon.exe (options/Enable Boot Logging).

Lors du redémarrage du micro-ordinateur, ce pilote est actif très tôt et peut ainsi capturer tous les événements du démarrage.

On peut l'arrêter la capture et sauvegarder les événements :

- de manière interactive avec l'interface graphique classique de Procmon,

- A VERIFIER :ou avec la ligne de commande suivante à placer dans un script par exemple :

```
C:\procmon\Procmon /BackingFile C:\procmon\log.pml /AcceptEula /Quiet /noconnect
```

☛ Pendant la capture, le pilote remplit un fichier temporaire situé dans le répertoire « ... \windows\procmon.PMB »

En fin de capture, lors de la sauvegarde, le pilote est supprimé. Parfois il reste présent. Et s'il reste actif lors du prochain boot dans le VHD, il produira un nouveau fichier qui pourrait remplir ce VHD.

Avec Winpe, il faut

- installer ce pilote en créant les entrées dans la ruche System,
- copier le pilote Procmon23.sys dans le répertoire « ...windows\system32\drivers ».

Pour disposer du pilote, il faut lancer une capture avec Promon et recopier le fichier dans un répertoire de sauvegarde.

Pour disposer des entrées nécessaires à créer dans la ruche de Winpe, il est plus simple de mettre en oeuvre cette capture et de faire un "export" avec regedit.

Dans le cas où Winpe est utilisé en mode Flat avec un disque virtuel VHD, et pour des tests successifs, il faut vérifier et recopier à nouveau le pilote dans ...windows\system32\drivers.



Astuce mise en oeuvre pour arrêter la capture ce procmon : un cmd est lancé avec Winpeshl. Ce cmd lance le programme « timeout.exe ». A la fin de ce délai, on lance la commande d'arrêt de la capture.

BSOD : l'avantage du VHD

Le VHD permet de générer un fichier de Dump que l'on peut ensuite analyser avec Windbg.

Si mes observations sont correctes, le fichier de dump est créé lors du boot suivant. L'Os redémarre et exploite le fichier « pagefile.sys » qui contient les détails de l'erreur. Et si l'on ne veut ou ne peut rebooter, il est possible de prendre le fichier « pagefile.sys » et de l'analyser avec WinDbg.

Windbg : trop d'informations à écrire pour l'instant

symboles

Kernel debugger network

Kernel debugger hyperV

Stratégies d'investigation et de construction

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.

Pour une meilleure efficacité, il est possible de réaliser des captures des modifications faites par l'installation d'un composant. Mais ce n'est pas toujours possible.

Cette recherche du détail devient vite fastidieuse.

De plus, quand on a ajouté beaucoup de composants, la taille de la ruche « software » ainsi complétée est presque égale à celle de la référence.

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. Dans cet environnement Winpe, les ACL ne posent pas de soucis, ne génère pas d'erreurs.

Il ne faut pas oublier ensuite de remplacer les valeurs inappropriées comme les chemins par exemple, « C:\ » à remplacer par « X:\ ».

Attention à la ruche "currentuser".

La totalité de la ruche «Software» : seul cas traité par mes scripts !

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».

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

☛ Un écueil déjà rencontré : j'ai parfois été obligé de remettre la clé « sideByside » de Winpe dans la ruche « software » provenant de Install.Wim car les versions de Winpe de l'ADK ne sont pas toujours actualisée avec l'ISO de MS.


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.

 les clés System\control...\control\MiniNt et «SystemSetupInProgress» sont un indicateur, de l'activité d'un Os Winpe. Elles sont testées par exemple par eventvwr.msc, « remote powerhell», spooler, termsrv, dsmsrv...

La ruche «Drivers»

Pour faciliter les ajouts de pilotes et lever tous les obstacles, j'ai fini par utiliser la ruche « Drivers » de Install.Wim en totalité.

A VERIFIER : quel serait l' avantage de recopier aussi la clé :
HKEY_LOCAL_MACHINE\SYSTEM\DriverDatabase

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 trop de fichiers sont inutiles..

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. Il faut trouver le test, la comparaison qui donnera des informations pertinentes.

Migrer vers une nouvelle version d'ADK?

Lorsqu'une nouvelle version de WinPe est disponible, le Winpe construit sur les bases des anciennes versions de scripts de MicroWinpeBuilder a bien peu de chance de fonctionner du premier coup. Au démarrage, on constate parfois que l'écran devient noir et que tout semble figé ! En effet, de nouveaux fichiers/clés apparaissent et d'autres disparaissent dans la fourniture de MS. D'où la nécessité de disposer d'une méthode efficace pour identifier les divergences.

Bien sûr Procmon reste un outil efficace mais face à un écran noir, comment interagir avec ce programme qui ne fonctionne pas, comment le découvrir?

BSOD

Il est préférable d'utiliser un VHD pour bénéficier de l'analyse du fichier de dump.

Un écran tout noir : le pire

Se souvenir que la fonctionnalité Themes :

- affiche les barres de titre des fenêtres avec une nouvelle esthétique (coin carré ...)

- utilise les services CoreMessagingRegistrar et Themes

Lors du démarrage, Winlogon.exe met en oeuvre le mécanisme DWM et l'initialise avec dwminit.dll si ce fichier est présent.

En activant la capture au démarrage de Procmon, on arrive à identifier les composants utiles pour afficher le bureau.

Si tous les composants utiles pour DWM ne sont pas identifiés et disponibles, l'écran restera noir.

Ne pas oublier de vérifier la présence de « D2D1.dll.mui » dans le répertoire de langue : sinon écran noir.

On peut procéder par étapes, neutraliser dwminit.dll pour désactiver DWM et vérifier les services, puis remettre dwminit.dll.

Anomalie lors de l'ouverture de la session ADM

On peut vérifier que le mécanisme "mode console" fonctionne. Pour cela renommer Windows.ui.logon.dll.

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.

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. 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

ADK

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

ISO de windows10

L'image ISO de windows10 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.PS1

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 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

généré 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
- pour chaque composant, copies des fichiers
- Dism pour l'ajout de pilote

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

Les fichiers supplémentaires : ProductOptions et English,... TODO

ProductOptions

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

English

C'est la traduction du mode d'emploi.

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».

Get-checksumPrg

Il m'a été utile pour modifier fbWf.sys

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)

SearchSignature

J'avais écrit rapidement un bout de script en PS. Les fichiers à modifier étant de petite taille (moins de 500KB), le traitement avec Powershell convenait. Au fil des années, j'ai pris le temps d'inclure un peu de code C# dans le script PS pour accélérer le traitement.

Et comme maintenant j'ai choisi de modifier plusieurs programmes, il est intégré dans mon gros script. Mais j'en garde une version autonome pour faire quelques vérifications ponctuelles.

La dll de hook pour WPD/MTP

```
// dllmain.cpp : Defines the entry point for the DLL application.
```

```
#include "stdafx.h"
```

```
#include <windows.h>
```

```
#include <TIHelp32.h>
```

```
HANDLE OpenLog();
```

Email : noelblanc.Winpe@free.fr


```

void WriteLog(HANDLE hFile, TCHAR *text);
LSTATUS SimpleDelai();

// le main
BOOL APIENTRY DllMain(HMODULE hModule,
    DWORD ul_reason_for_call,
    LPVOID lpReserved)
{
    switch (ul_reason_for_call) {
    case DLL_PROCESS_ATTACH:
        SimpleDelai();
        break;
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}

DWORD FindAppProcessID(HANDLE hFile, TCHAR *strAppName)
{
    TCHAR buff[512] = { 0 };
    HANDLE handle = CreateToolhelp32Snapshot(TH32CS_SNAPALL, 0);
    PROCESSENTRY32 Info;
    Info.dwSize = sizeof(PROCESSENTRY32);
    if (Process32First(handle, &Info)) {
        do {
            TCHAR *ss = Info.szExeFile;
            wsprintf(buff, TEXT("%d:%s"), Info.th32ProcessID, ss);
            WriteLog(hFile, buff);
            if (wcscmp(ss, strAppName) == 0) {
                CloseHandle(handle);
                return Info.th32ProcessID;
            }
        } while (Process32Next(handle, &Info));
        CloseHandle(handle);
    }
    return 0;
}

// "winlogon" is loaded before "services".
// for winlogon only (in dllmain, reason == DLL_PROCESS_ATTACH):
// We kill the Minint key. We wait for svchost.exe process.
// after delay, we create the minint key

```

```

LSTATUS SimpleDelai()
{
    TCHAR szFileName[MAX_PATH] = { 0 };
    GetModuleFileName(NULL, szFileName, MAX_PATH);
    HKEY dmy;
    if (wcsncmp(szFileName, L"X:\\windows\\system32\\winlogon.exe", 32) == 0) {
        RegDeleteKey(HKEY_LOCAL_MACHINE, L"SYSTEM\\CurrentControlSet\\
Control\\MiniNT");
        TCHAR buff[512] = { 0 };
        HANDLE f = OpenLog();
        SYSTEMTIME sys;
        GetLocalTime(&sys);
        sprintf(buff, TEXT("%4d-%02d-%02d %02d:%02d:%02d.%03d"), sys.wYear,
sys.wMonth, sys.wDay, sys.wHour, sys.wMinute, sys.wSecond, sys.wMilliseconds);
        WriteLog(f, buff);
        for (int i = 0; i < 30; i++) {
            GetLocalTime(&sys);
            sprintf(buff, TEXT("%4d-%02d-%02d %02d:%02d:%02d.%03d"), sys.wYear,
sys.wMonth, sys.wDay, sys.wHour, sys.wMinute, sys.wSecond, sys.wMilliseconds);
            WriteLog(f, buff);
            if (FindAppProcessID(f, TEXT("svchost.exe")) > 0) break;
            Sleep(500);
            RegCreateKey(HKEY_LOCAL_MACHINE, L"SYSTEM\\CurrentControlSet\\
Control\\MiniNT", &dmy);
            CloseHandle(f);
        }
        return ERROR_SUCCESS;
    }
}

HANDLE OpenLog() {
    HANDLE hFile = 0;
    TCHAR szFileName[MAX_PATH] = { 0 };
    hFile = CreateFile(L"X:\\hook.dat", FILE_APPEND_DATA, FILE_SHARE_READ |
FILE_SHARE_WRITE,
        NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    return hFile;
}

void WriteLog(HANDLE hFile, TCHAR *text)
{
    SetFilePointer(hFile, 0, NULL, FILE_END);
    DWORD dwWritten;
    TCHAR buff[512] = { 0 };
    sprintf(buff, TEXT("%s\r\n"), text);
}

```

```
    if (hFile) {  
        WriteFile(hFile, buff, lstrlenW(buff) * sizeof(TCHAR), &dwWritten, NULL);  
        FlushFileBuffers(hFile);  
    }  
}
```

ARCHIVES

ARCHIVES

DCOM et la bascule vers l'interface graphique classique

Ne fonctionne plus depuis V1703

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

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

Si j'ai bien compris l'histoire de cette fonctionnalité, ces deux objets n'étaient plus créés avec certaines versions de Winpe.

Des développeurs chinois avaient donc écrit les logiciels « SetWow64.exe » et « LoadWoW64.exe ». Ces logiciels sont repris par l'équipe de WinPeSe.

Or, depuis peu (Version 1709), Winpe assure la création de ces deux objets System. Donc ces logiciels ne sont plus utiles.

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 et Procexp.exe

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

Il me semble qu'il ne donne pas de bons résultats en 64 bits.

Procexp avec «Ctrl+H» me semble plus efficace. Mais il ne donne pas un aperçu global.

Script MonSetWow64 pour le sous-système 32 bits

Devenu inutile au moins depuis V1709

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 de systinternals n'existe pas en version 64bits. Un autre programme du même nom existe mais je ne souviens pas du site où il est disponible.

Création

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

Modification du wallpaper et de themecpl.dll

Ne fonctionne plus depuis V1703

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 :

Email : noelblanc.Winpe@free.fr

"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 intéressent.

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 écran

Le second écran 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.

Voir mon script «modify-themecpl.PS1»

Win10pCap and Wireshark hors Boot.Wim : A REVOIR



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