LA COMPATIBILITÉ À LA RESCOUSSE

Cédric PERNET – cedric.pernet@gmail.com

mots-clés : BASE DE REGISTRE / EXÉCUTION / ANALYSE

Le domaine de l'inforensique est en perpétuelle évolution. Les systèmes d'exploitation changent, sont mis à jour, sont modifiés. Il en va de même pour les logiciels et le comportement des usagers. Cet ensemble en constante mouvance génère une charge de travail importante pour l'investigateur numérique, qui doit perpétuellement se maintenir à jour de ses connaissances.

Parmi ces dernières, l'une des plus importantes est celle des différentes ruches de la base de registre des systèmes d'exploitation Windows. Elle constitue l'un des viviers les plus intéressants pour l'analyste, mais probablement le plus dense et le plus complexe.

À l'intérieur de cette base se trouvent des milliers d'entrées, certaines plus obscures que d'autres. Parmi ces dernières, il en existe un certain nombre permettant de savoir si un binaire a été exécuté ou non. L'AppCompatCache en est un, qui n'est pas forcément très connu car peu documenté.

1 Indicateurs d'exécution

De nombreux indicateurs permettent de déterminer si un binaire a été exécuté ou non sur un système Windows.

Ces entrées sont connues de la plupart des investigateurs numériques, mais connaître ne signifie pas forcément utiliser et exploiter judicieusement.

Voici un bref rappel des principales entrées indicatrices d'exécution de binaires :

- Fichiers Prefetch

Il s'agit de fichiers localisés dans un répertoire *Prefetch* du répertoire d'installation de Windows. Ces fichiers fournissent des informations sur les derniers fichiers exécutés sur le système ainsi qu'un horodatage de la dernière exécution. Il faut cependant rappeler ici que l'activation de ces fichiers n'est pas mise en place par défaut sur les serveurs Windows et que les prefetchs sont désactivés par défaut par Windows s'il est lancé d'un disque dur SSD (ce qui est discutable, mais là n'est pas le sujet).

- Fichier d'hibernation

`hiberfil.sys` est un fichier localisé à la racine de la partition contenant le système d'exploitation Windows. Ce fichier est une copie compressée de la mémoire vive produite lors d'une mise en hibernation du système. Une fois décompressé **[1]**, ce *dump* de la RAM peut être analysé avec des outils tels que Volatility **[2]** afin d'examiner les processus en fonctionnement au moment du dump.

- Base de registre : MUICache

À chaque exécution d'un programme par un utilisateur, Windows stocke le nom de l'application dans une clé de la base de registre appelée MUICache. Le défaut de cette clé est qu'elle ne contient aucun horodatage hormis celui de la dernière mise à jour de la clé elle-même, mais les informations que cette clé contient sont néanmoins intéressantes. Elles peuvent notamment être croisées avec d'autres sources afin de compléter une analyse inforensique. Cette clé particulière peut être parcourue avec MUICacheView **[3]** notamment.

- **Base de registre : UserAssist**

  La clé `UserAssist` (localisée dans `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\`) contient un ensemble de sous-clés permettant de retracer l'activité de l'utilisateur **[4]**, dans une certaine mesure. Cette clé présente beaucoup d'intérêt pour un analyste inforensique. Elle permet notamment de connaître le nombre d'exécutions d'une application (par utilisateur) et présente un horodatage de la dernière exécution de chacun de ces programmes. À noter que la structure de cette clé n'est pas la même sous Windows XP et sous Windows 7 et supérieur.

- **Base de registre : RecentDocs**

  La base de registre contient une clé `RecentDocs` (localisée dans `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\`) qui permet d'obtenir un historique des fichiers accédés récemment par l'utilisateur. Ces fichiers sont classés par sous-clé correspondant aux extensions utilisées : jpg, mp3, doc, etc. Le *plugin* « recentdocs » de RegRipper **[5]** permet de parcourir cette structure facilement. Une limitation est à noter au niveau de l'horodatage : il ne contiendra que les dernières dates de modification de chaque sous-clé.

- **Base de registre : RunMRU**

  Cette clé, localisée dans `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\`, permet d'obtenir l'historique des commandes effectuées à partir de la boîte « Run » du menu « Démarrer » de Windows. Elle présente elle aussi une limitation importante : seule la dernière date d'utilisation est indiquée dans la base. Le plugin `runmru.pl` **[6]** de RegRipper permet de parcourir cette clé confortablement.

- **Base de registre : AppCompatFlags**

  Cette clé de la base de registre (emplacement : `NTUSER.DAT\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags`) permet d'obtenir la liste des fichiers exécutés configurés pour être lancés en mode compatibilité. Seul horodatage : la date de la dernière modification de la clé. Elle peut être parcourue avec le plugin `appcompatflags.pl` **[7]** de RegRipper.

- **Emplacements d'exécution automatique**

  Il en existe un grand nombre, localisés principalement en base de registre (les fameuses clés « Run »). Mais il ne faudrait pas oublier non plus les services démarrés, que nous pouvons voir dans le journal d'événements « System ». Les ID d'événements 7034, 7035, 7036, et 7040 **[8]** nécessitent une attention particulière. Pour ce qui est des clés Run, l'outil de référence pour les parcourir est AutoRuns **[9]**.

- **Jump Lists**

  Les « jump lists » **[10]** sont des *items* qui ont fait leur apparition sur les systèmes Windows 7 et 8. Pour simplifier, il s'agit d'ajouts aux barres de tâches spécifiques à chaque application, accessibles par clic droit dans l'interface de Windows, permettant notamment d'avoir les historiques récents. Les *jump lists* permettent ainsi d'obtenir des informations sur l'exécution d'applications.

Cette liste n'est pas exhaustive. D'autres entrées moins connues existent, certaines étant spécifiques à des applications par exemple. Parmi ces entrées moins connues existe ce que l'on appelle l'« AppCompatCache ».

2 AppCompatCache

2.1 Historique

Cet artefact inforensique a été découvert par Andrew Davis **[11]** en avril 2012. Il était jusqu'alors passé inaperçu et ne faisait l'objet d'aucun traitement particulier par la communauté des investigateurs numériques, du moins publiquement. Quelques personnes se sont posé la question de ce qu'était cette clé de la base de registre dès 2009, mais sans s'en préoccuper plus que ça, et sans obtenir de réponse, même sur des forums internationaux dédiés à l'inforensique.

Cette découverte d'Andrew Davis fut immédiatement suivie par la publication d'un outil **[12]** libre pour pouvoir parcourir cette structure, sur lequel nous reviendrons ultérieurement.

Le jour suivant l'annonce de cette découverte, Harlan Carvey publiait un module `appcompatcache.pl` **[13]** pour RegRipper.

2.2 Description

L'AppCompatCache est une clé de la base de registre des systèmes Windows, localisée dans la ruche SYSTEM, présente à partir des systèmes d'exploitation XP et supérieurs.

Cette clé est générée par la base de données de compatibilité d'applications Windows **[14]**. Ces données cache permettent au système d'identifier des applications présentant des problèmes de compatibilité.

L'AppCompatCache est implémenté de façon différente en fonction du système d'exploitation.

2.2.1 Windows XP (32 bits)

Sous Windows XP 32 bits, la clé se trouve à l'emplacement suivant : `HKLM\SYSTEM\CurrentControlSet\Control\SessionManager\AppCompatibility\`

`AppCompatCache`

Ce cache est décrit par la société Mandiant sous la forme d'une structure avec un en-tête de 400 octets démarrant par « 0xDEADBEEF ». Cet en-tête contient le nombre d'entrées ainsi que les index utilisés par le gestionnaire de cache. Chacune de ces entrées est une structure contenant le chemin complet vers le binaire exécuté, un horodatage de la dernière modification, la taille du binaire, ainsi qu'un horodatage de la dernière modification de l'entrée.

```
typedef struct AppCompatCacheEntry_XP{
WCHAR Path[MAX_PATH+4];
FILETIME ftLastModTime;
LARGE_INTEGER qwFileSize;
FILETIME ftLastUpdateTime;
} APPCOMPATCACHE_ENTRY32_XP;
```

Windows XP 32 bits peut stocker un maximum de 96 entrées de ce type dans la clé **AppCompatCache**. Ces entrées sont ajoutées à la clé dans les deux conditions suivantes :

- Les métadonnées d'un fichier ont changé depuis la dernière exécution, le fichier venant d'être ré-exécuté.
- Un nouveau fichier est exécuté.

La date de dernière modification de l'entrée est un bon indicateur pour un enquêteur souhaitant connaître le moment précis de la dernière exécution d'un fichier se trouvant dans ce cache. Il est cependant plus confortable de toujours confirmer cette date par d'autres éléments de la base de registre (UserAssist par exemple).

2.2.2 Windows XP (64 bits) + Windows Server 2003

L'emplacement de l'AppCompatCache sur ces systèmes a été déplacé vers : **HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache**

AppCompatCache

La structure de l'AppCompatCache a beaucoup changé sur ces systèmes. Pour ce qui est des en-têtes, l'ancien « 0xDEADBEEF » des systèmes Windows XP 32 bits a été remplacé par « 0xBADC0FEE », suivi du nombre d'entrées. À noter que la date de dernière mise à jour présente dans XP 32bits n'est plus présente.

La structure de chaque entrée devient :

- Version Windows 2003 32 bits :

```
//32-bit Win2k3 AppCompatCache Structure
typedef struct AppCompatCacheEntry32_2k3 {
USHORT wLength;
USHORT wMaximumLength;
DWORD dwPathOffset;
FILETIME ftLastModTime;
LARGE_INTEGER qwFileSize;
} APPCOMPATCACHE_ENTRY32_2k3;
```

- Version Windows 2003 64 bits et Windows XP 64 bits :

```
//64-bit Win2k3 AppCompatCache Structure
typedef struct AppCompatCacheEntry64_2k3 {
USHORT wLength;
USHORT wMaximumLength;
DWORD dwPadding;
QWORD dwPathOffset;
FILETIME ftLastModTime;
LARGE_INTEGER qwFileSize;
} APPCOMPATCACHE_ENTRY64_2k3;
```

Sur ces systèmes, le cache peut compter jusqu'à 512 entrées.

Les nouvelles entrées sont créées :

- lorsqu'un fichier est exécuté et que ses métadonnées sont mises à jour ;
- si un même fichier est exécuté à partir d'un autre répertoire.

2.2.3 Windows Server 2008 / Windows Vista

L'AppCompatCache se trouve ici pour ces systèmes :

HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache

AppCompatCache

Il n'y a pas de changement au niveau des en-têtes par rapport à Windows Server 2003, par contre la structure des entrées est légèrement différente :

- Version Windows Server 2008/Vista 32 bits :

```
//32-bit Vista/2k8 AppCompatCache Entry Structure
typedef struct AppCompatCacheEntry32_Vista {
USHORT wLength;
USHORT wMaximumLength;
DWORD dwPathOffset;
FILETIME ftLastModTime;
DWORD dwInsertFlags;
DWORD dwFlags;
} APPCOMPATCACHE_ENTRY32_VISTA;
```

- Version Windows Server 2008/Vista 64 bits :

```
//64-bit Vista/2k8 AppCompatCache Entry Structure
typedef struct AppCompatCacheEntry64_Vista {
USHORT wLength;
USHORT wMaximumLength;
DWORD dwPadding;
QWORD dwPathOffset;
FILETIME ftLastModTime;
DWORD dwInsertFlags;
DWORD dwFlags;
} APPCOMPATCACHE_ENTRY64_VISTA;
```

Certains binaires présents dans ce cache ne sont pas exécutés. Ils s'y retrouvent suite à une inscription par **explorer.exe**, qui ajoute les métadonnées des binaires d'un répertoire quand il le parcourt.

Le drapeau « dwInsertFlags » devient du coup particulièrement intéressant. Il semblerait que lorsqu'il est défini, il indiquerait une exécution de binaire. Le conditionnel utilisé est basé sur le fait que pour le moment personne n'a découvert de cas contradictoire.

Enfin, le cache peut compter jusqu'à 1024 entrées sur ces systèmes.

L'en-tête de l'AppCompatCache démarre par « 0xBADC0FEE », suivi par le nombre d'entrées du cache, puis des données statistiques sur le cache maintenues par le noyau Windows. Suivent les entrées, qui présentent la structure suivante :

- Windows 2008 R2 et Windows 7 32bits :

```
//32-bit Win7/2k8R2 AppCompatCache Entry Structure
typedef struct AppCompatCacheEntry32_Win7{
USHORT wLength;
USHORT wMaximumLength;
DWORD dwPathOffset;
FILETIME ftLastModTime;
DWORD dwInsertFlags;
DWORD dwShimFlags;
DWORD dwBlobSize;
DWORD dwBlobOffset;
} APPCOMPATCACHE_ENTRY32_WIN7;
```

- Version 64 bits :

```
//64-bit Win7/2k8R2 AppCompatCache Entry Structure
typedef struct AppCompatCacheEntry64_Win7{
USHORT wLength;
USHORT wMaximumLength;
DWORD dwPadding;
QWORD dwPathOffset;
FILETIME ftLastModTime;
DWORD dwInsertFlags;
DWORD dwShimFlags;
QWORD qwBlobSize;
QWORD qwBlobOffset;
} APPCOMPATCACHE_ENTRY64_WIN7;
```

Le drapeau « dwInsertFlags » permet à nouveau de déterminer si un binaire a été exécuté ou ajouté au cache par `explorer.exe`

« dwShimFlags » est relatif à la base de données de compatibilité et peut être utilisé par `apphelp.dll`

Quant à « dwBlobSize » et « dwBlobOffset », leur utilisation demeure inconnue, ils sont généralement à zéro. BlobOffset est un *offset* vers une structure opaque contenue dans les données cache, de taille BlobSize. Il a néanmoins été observé que des valeurs y étaient inscrites lors de l'installation de logiciels par certains *installers*.

Le cache pour ces systèmes peut contenir un maximum de 1024 entrées.

L'emplacement du cache n'a pas changé par rapport à Vista, il se trouve dans la ruche SYSTEM :

`HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache\`

`AppCompatCache`

Plusieurs outils gratuits et libres sont à notre disposition pour parcourir efficacement les entrées de l'AppCompatCache, les deux principaux étant ShimCacheParser et un module de RegRipper.

Il s'agit du script Python d'Andrew Davis. Ce script **[12]** peut se lancer sur un système local ou sur une ruche d'un autre système, sur une extraction de l'AppCompatCache, ou encore sur des extraits MIR, l'outil de réponse à incident de Mandiant.

Le résultat peut être obtenu directement en console, ou sous forme de fichier CSV, plus pratique à parcourir.

Pour rappel, le champ « Process Exec Flag » n'existe pas sous Windows XP 32bits et nous n'obtenons donc aucun résultat sur ce système.

	A	B	C	D	E
1	Last Modified	Last Update	Path	File Size	Process Exec Flag
2	11/18/05 12:54:34	04/11/2013 09:07	C:\Program Files\Fichiers communs\VMware\VMware Virtual Image Editing\vmount2.exe	245760	N/A
3	10/14/09 14:58:02	05/13/13 10:04:59	C:\Program Files\PE Explorer\pexplorer.exe	3007224	N/A
4	04/21/12 01:16:21	02/20/13 16:19:52	C:\Program Files\Mozilla Firefox\firefox.exe	924600	N/A

Fig 1 : Exemple de résultat obtenu avec ShimCacheParser.py, pour un système Windows XP 32 bits

3.2 RegRipper / Rip

RegRipper **[15]** est un outil permettant de travailler confortablement sur les différentes ruches des systèmes d'exploitation Windows. Sa version en ligne de commandes s'appelle rip, et se base sur de nombreux *plugins*. L'un de ses plugins se nomme « appcompatcache » et permet d'obtenir le contenu de l'AppCompatCache d'une ruche SYSTEM :

```
c:\TOOLS\RR>rip -r c:\copy_of_system_hive -p appcompatcache > c:\
Users\REM\Desktop\results_appcompatcache.txt
Launching appcompatcache v.20130425
```

Nous redirigeons le résultat vers un fichier, qui nous présente le début de contenu suivant :

```
appcompatcache v.20130425
(System) Parse files from System hive Shim Cache

Signature: 0xbadc0fee
Win2K8R2/Win7, 64-bit

C:\Program Files (x86)\Windows Media Player\wmplayer.exe
ModTime: Sun Nov 21 03:25:10 2010 Z

C:\Windows\system32\mspaint.exe
ModTime: Tue Jul 14 01:39:24 2009 Z
```

Le module « appcompatcache » nous indique sa version, suivi de la signature de début d'en-tête « 0xBADC0FEE ». Il nous fournit ensuite le système d'exploitation, puis liste toutes les entrées.

Conclusion

Cet article s'est focalisé sur les aspects inforensiques exploitables, sans entrer dans le détail de toutes les structures et implémentation complète par l'« Application Compatibility Database ». Pour plus de détails sur ces aspects, il est recommandé de lire le document de référence d'Andrew Davis référencé précédemment ainsi qu'une série de billets du blog d'Alex Ionescu **[16]**.

Les données qu'il est possible de collecter dans l'AppCompatCache varient grandement d'un système à un autre pour un analyste inforensique. Alors que sous un système Windows XP 32 bits il est possible d'obtenir un horodatage de la dernière mise à jour d'une entrée, cela n'est plus possible sur un système tel que Windows 7 par exemple. De même, la taille exacte d'un binaire ne pourra pas forcément être obtenue par l'AppCompatCache.

Les données de ce cache sont néanmoins très intéressantes. Il s'agit d'un moyen de plus de corréler de l'information par rapport à d'autres indicateurs d'exécution. De plus, il peut arriver dans certaines circonstances que l'AppCompatCache soit le seul indicateur de l'exécution d'un fichier.

L'AppCompatCache est donc un élément important qui devrait être systématiquement analysé lorsqu'une investigation porte sur des exécutions de fichiers. ■

■ REMERCIEMENTS

Cet article se base fortement sur la seule ressource complète sur le sujet actuellement, le document d'Andrew Davis, que je remercie.

Je remercie également CASSIDIAN CyberSecurity, en particulier David Bizeul, Jérôme Leseinne et Fabien Perigaud. Un grand merci également à Eric Freyssinet, Philippe Teuwen, Benjamin Caillat pour leur relecture attentive, ainsi qu'à Boris le Zombie pour sa formidable inertie.

■ RÉFÉRENCES

[1] par exemple avec http://www.moonsols.com/windows-memory-toolkit/

[2] https://www.volatilesystems.com/default/volatility

[3] MUICacheView – http://www.nirsoft.net/utils/muicache_view.html

[4] UserAssist Forensics (timelines, interpretation, testing, & more) - http://www.4n6k.com/2013/05/userassist-forensics-timelines.html

[5] recentdocs.pl - https://code.google.com/p/regripperplugins/source/browse/recentdocs.pl

[6] runmru.pl - https://code.google.com/p/regripperplugins/source/browse/runmru.pl

[7] appcompatflags.pl - https://code.google.com/p/regripperplugins/source/browse/appcompatflags.pl

[8] Service Events Logging - http://technet.microsoft.com/en-us/library/dd349442%28v=ws.10%29.aspx

[9] AutoRuns for Windows - http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx

[10] Jump List Forensics: AppIDs Part 1 - http://www.4n6k.com/2011/09/jump-list-forensics-appids-part-1.html

[11] Leveraging the Application Compatibility Cache in Forensic Investigations - http://www.mandiant.com/library/Whitepaper_ShimCacheParser.pdf

[12] ShimCacheParser.py - https://github.com/mandiant/ShimCacheParser

[13] appcompatcache.pl - https://code.google.com/p/regripperplugins/source/browse/_old_/appcompatcache.pl?r=beb002fcc6c78d21331b202288764f347c39eaeb

[14] Windows Application Compatibility Database - http://msdn.microsoft.com/en-us/library/bb432182%28v=vs.85%29.aspx

[15] RegRipper - https://code.google.com/p/regripper/

[16] Secrets of the Application Compatilibity Database (SDB) – Part 1 - http://www.alex-ionescu.com/?p=39