Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PREMIER MINISTRE

Secrétariat général
de la défense
et de la sécurité nationale

*Agence nationale de la sécurité
des systèmes d'information*

Paris, le 18 juillet 2016

N° DAT-NT-008/ANSSI/SDE/NP

Nombre de pages du document
(y compris cette page) : 31

Note technique

Recommandations de sécurité relatives aux environnements d'exécution Java sur les postes de travail Microsoft Windows

Public visé :

Développeur	✓
Administrateur	✓
RSSI	
DSI	
Utilisateur	

Informations

Avertissement

Ce document rédigé par l'ANSSI présente les « **Recommandations de sécurité relatives aux environnements d'exécution Java sur les postes de travail Microsoft Windows** ». Il est téléchargeable sur le site www.ssi.gouv.fr. Il constitue une production originale de l'ANSSI. Il est à ce titre placé sous le régime de la « Licence ouverte » publiée par la mission Etalab (www.etalab.gouv.fr). Il est par conséquent diffusable sans restriction.

Ces recommandations sont livrées en l'état et adaptées aux menaces au jour de leur publication. Au regard de la diversité des systèmes d'information, l'ANSSI ne peut garantir que ces informations puissent être reprises sans adaptation sur les systèmes d'information cibles. Dans tous les cas, la pertinence de l'implémentation des éléments proposés par l'ANSSI doit être soumise, au préalable, à la validation de l'administrateur du système et/ou des personnes en charge de la sécurité des systèmes d'information.

Personnes ayant contribué à la rédaction de ce document :

Contributeurs	Rédigé par	Approuvé par	Date
Division Assistance Technique, Bureau Audits et Inspections	DAT	SDE	18 juillet 2016

Évolutions du document :

Version	Date	Nature des modifications
1.0	4 mars 2016	Version initiale
2.0	18 juillet 2016	Intégration Java Web Start et obsolescence du module complémentaire NPAPI

Pour toute question :

Contact	Adresse	@mél
Division Assistance Technique de l'ANSSI	51 bd de La Tour-Maubourg 75700 Paris Cedex 07 SP	conseil.technique@ssi.gouv.fr

Table des matières

1 Préambule 4

2 Présentation 4

2.1 Risques de sécurité des JRE . . . 4
2.2 Java, JRE, et JavaScript . . . 5
2.3 Les différentes versions de JRE et leur cohabitation . . . 5
2.4 Types d'applications Java et modules complémentaires . . . 6
2.4.1 Applications lourdes . . . 6
2.4.2 Applications légères . . . 6
2.4.3 Applications Java Web Start . . . 6
2.5 Java Web Start et JNLP . . . 6
2.6 NPAPI et le module complémentaire Java . . . 7

3 La phase d'inventaire 8

3.1 Recensement des applications Java et de leurs utilisateurs . . . 8
3.2 Recensement des postes munis de JRE et des versions déployées . . . 8
3.3 Pertinence de la présence de JRE . . . 8

4 La démarche de sécurisation 9

4.1 Interdiction des JRE inutiles . . . 9
4.2 Désactivation des modules complémentaires Java et des Java Web Start inutiles . . . 9
4.3 Remplacement des appliquettes Java . . . 11
4.3.1 Ré-écriture d'une appliquette Java en applications Java Web Start . . . 11
4.3.2 Migration simple des appliquettes Java en applications Java Web Start . . . 11
4.4 Configuration centralisée . . . 12
4.5 Télédéploiement du JRE . . . 13
4.6 Gestion des mises à jour . . . 14

5 Pour aller plus loin 15

5.1 Cas particulier des systèmes d'exploitation Microsoft Windows 64 bits . . . 15
5.2 Fichiers `.jar` et `.jnlp` . . . 16
5.3 Navigateurs multiples . . . 16
5.4 Intégrer le MCO et le MCS dans les contrats . . . 17

A Scripts d'inventaire des JRE 18

A.1 Script d'inventaire en PowerShell . . . 18
A.2 Script d'inventaire sur page Web . . . 19

B Configuration centralisée par fichiers 21

B.1 Création du fichier `deployment.properties` . . . 21
B.2 Création et télédéploiement du fichier `deployment.config` . . . 23

C Télédéploiement du JRE 8 par GPO 26
C.1 Téléchargement du JRE 26
C.2 Déploiement par MSI 26
C.3 Déploiement par exécutable 28

1 Préambule

La technologie Java, rachetée en 2009 par Oracle Corporation, reste aujourd'hui très répandue et utilisée par de nombreuses applications. Ces applications peuvent se présenter sous la forme d'appliquettes [1] exécutées depuis des clients légers (navigateurs Web) ou bien d'applications lourdes installées sur les postes utilisateurs ou téléchargées à l'exécution par Java Web Start via JNLP [2]. L'exécution de ces applications nécessite l'installation préalable des environnements d'exécution Java (appelés JRE, acronyme de *Java Runtime Environment*) sur les postes utilisateurs, on trouve par conséquent ces JRE souvent déployés dans les environnements professionnels.

Comme tout composant logiciel utilisé pour la navigation Web, ces environnements d'exécution Java sont une cible privilégiée des attaquants, ils font régulièrement l'actualité informatique de par les vulnérabilités fréquentes qui les touchent ainsi que leur exploitation facile et massive (voir section 2.1). Qui plus est, les appliquettes Java ont une obsolescence annoncée qu'il convient d'anticiper tant par les développeurs d'applications Java que par les équipes d'administration des systèmes d'information.

Après une présentation des problèmes de sécurité liées à l'utilisation des environnements d'exécution Java au sein d'un système d'information, ce document propose une méthodologie d'inventaire destinée à recenser précisément les besoins liés à Java, puis des recommandations pour mener une démarche de sécurisation de ces environnements.

2 Présentation

2.1 Risques de sécurité des JRE

Les JRE font souvent l'objet de vulnérabilités, corrigées tardivement, et souvent exploitées avant leur publication et leur correction (codes d'exploitation dits *zero-day*). Leur mise à jour n'est qui plus est pas nécessairement automatique, il est alors fréquent que des versions vulnérables du JRE soient à l'origine de la compromission d'un poste utilisateur, et éventuellement d'un système d'information par rebond.

Ajoutons à cela le fait que certaines applications ne sont compatibles qu'avec une version bien précise de l'environnement d'exécution Java (c'est à dire ancienne et vulnérable dans la majorité des cas), imposant donc la cohabitation potentielle de plusieurs versions du JRE sur un même poste utilisateur, et donc la présence de vulnérabilités non corrigées. La sécurité des JRE est donc un problème de taille pour le maintien en conditions de sécurité du système d'information, et qui l'expose à des risques importants. Force est toutefois de constater que ces risques sont parfois mal identifiés, et donc mal appréhendés.

Il est important de prendre conscience de l'inefficacité de l'antivirus face à ce type de menaces. À titre d'exemple, un code d'exploitation utilisant la vulnérabilité Java du 27 août 2012 [3] n'était identifié et bloqué que par moins de la moitié des logiciels antivirus du marché lors de sa parution. Il est également important de rappeler que les outils de réduction des risques de sécurité ou de protection contre les *zero-day* tels qu'EMET (*Enhanced Mitigation Experience Toolkit*) peuvent être d'une efficacité limitée face à ces menaces. En effet, les vulnérabilités Java consistent la plupart du temps en des échappements de bac à sable, n'utilisant pas de vulnérabilités de la JVM (Machine virtuelle Java) proprement dite, et ne font donc pas appel à des techniques contre lesquelles ces outils peuvent protéger.

1. Plus connues sous leur nom anglais *applets* et également francisées sous le terme *applettes*.
2. JNLP *Java Network Launching Protocol* est un protocole de démarrage d'application Java délivrée par réseau.
3. Voir http://www.cert.ssi.gouv.fr/site/CERTA-2012-ALE-002.

De multiples avis et de sécurité et bulletins d'actualité relatifs aux vulnérabilités Java peuvent être consultés sur le site du CERT-FR [4] :

- CERTFR-2016-AVI-027 ;
- CERTFR-2016-AVI-049 ;
- CERTFR-2016-AVI-108 ;
- CERTFR-2016-AVI-136 ;
- etc.

Plus particulièrement, y figurent les alertes critiques de sécurité du 27 août 2012 [5], du 10 janvier 2013 [6] et du 13 juillet 2015 [7] relatives à des vulnérabilités de Java massivement exploitées.

2.2 Java, JRE, et JavaScript

Attention à ne pas confondre Java et JavaScript, qui sont deux langages bien distincts et sans lien entre eux.

- Java est un langage compilé qui nécessite l'installation d'un environnement d'exécution (JRE), ainsi que des modules complémentaires (voir section 2.4) dans les navigateurs Web pour l'exécution des appliquettes ;
- JavaScript est un langage interprété et exécuté par un moteur JavaScript intégré dans les navigateurs, et très largement utilisé par les sites Internet. Celui-ci ne fait pas partie du périmètre de la présente note technique et ne présente actuellement pas le même niveau de risques.

2.3 Les différentes versions de JRE et leur cohabitation

Une version du JRE se présente sous la forme `JRE 1.X.Y_Z`, où :

- X est la version majeure (1.7 pour le JRE 7, ou 1.8 pour le JRE 8) ;
- Y est la version mineure (1.8.0 par exemple) ;
- Z est la version de mise à jour (1.8.0_92 pour la version `Update 92` du JRE 8, généralement incrémentée lorsqu'un correctif de sécurité est publié.

Plusieurs versions du JRE peuvent cohabiter sur un même poste de travail. L'installation d'une version du JRE antérieure à 1.6.0_10 inclus n'écrase pas les versions précédentes déjà installées. Pour les versions plus récentes, par défaut, l'installation d'une version mineure ou de mise à jour plus récente écrase le JRE de même version majeure déjà installé. Par exemple, le JRE 1.8.0_92 remplace le 1.8.0_91. Par contre il ne remplacera pas le 1.7.0_101. Il est donc peu probable que plusieurs JRE de même version majeure cohabitent sur un poste de travail, à moins que cela n'ait été fait volontairement en modifiant les dossiers d'installation par défaut. Il est en revanche fréquent de rencontrer des postes utilisateurs sur lesquels de multiples versions majeures du JRE sont installées.

Une appliquette peut également demander à s'exécuter sur une version particulière du JRE. Une appliquette malveillante peut par exemple choisir de s'exécuter sur une version 1.5 vulnérable (à condition bien entendu qu'elle soit installée) malgré la présence d'une version 1.8 à jour. Depuis le JRE 1.6.0_10, ce mécanisme de sélection a été modifié afin de faire intervenir une boite de dialogue d'avertissement, demandant à l'utilisateur d'autoriser ou non l'exécution de l'appliquette sur une version antérieure du JRE. Il reste néanmoins probable que l'utilisateur accepte l'exécution d'une JRE vulnérable sans mesurer l'impact de ce choix.

4. Se reporter à http://www.cert.ssi.gouv.fr/.
5. Se reporter à http://www.cert.ssi.gouv.fr/site/CERTA-2012-ALE-002.
6. Se reporter à http://www.cert.ssi.gouv.fr/site/CERTA-2013-ALE-001.
7. Se reporter à http://cert.ssi.gouv.fr/site/CERTFR-2015-ALE-007.

2.4 Types d'applications Java et modules complémentaires

2.4.1 Applications lourdes

Les applications lourdes sont exécutées sur les postes utilisateurs et requièrent uniquement l'installation préalable d'un JRE. Ces applications ne sont pas nécessairement installées sur les postes puisqu'elles sont la plupart du temps portables. Par contre les fichiers de l'application doivent être présents sur le poste ou accessibles via un partage réseau.

2.4.2 Applications légères

Les applications légères sont mises à disposition par des serveurs d'applications Web sous forme d'appliquettes et on y accède via le navigateur côté client. Elles nécessitent un JRE ainsi qu'un module complémentaire (également appelé *Plug-In* ou *Add-On*) qui vient se greffer au navigateur pour les exécuter. Les appliquettes Java sont entièrement téléchargées à chaque exécution, ce qui est loin d'offrir des performances optimales. Elles n'offrent par ailleurs pas toutes les fonctionnalités d'une application lourde et présentent certaines limitations [8]. Le risque de sécurité provient majoritairement de l'utilisation de ce module complémentaire lors de la navigation. Les serveurs d'applications utilisés sont généralement internes au système d'information mais peuvent tout aussi bien être sur Internet ou en extranet, ce qui a son importance dans la démarche de sécurisation.

Comme expliqué à la section 2.6, les appliquettes ne peuvent plus être exécutées dans la plupart des navigateurs maintenus du marché, à l'exception de Microsoft Internet Explorer 11.
Enfin, il convient de noter qu'Oracle a annoncé que les appliquettes Java seront considérées comme obsolètes à partir de la version 9 de Java. Elles sont donc destinées à disparaitre des versions futures de Java.

2.4.3 Applications Java Web Start

Une application utilisant Java Web Start (fonctionnalité apparue dans la version 1.4 du JRE) est un intermédiaire entre une application lourde et une application légère. En effet, à la différence des appliquettes, les applications Java Web Start n'ont pas de fonctionnalités limitées et ne requièrent pas de navigateur web pour leur exécution. À la différence des applications lourdes, elles peuvent être démarrées et téléchargées depuis une session Web ou un fichier de raccourci à l'aide de JNLP (mécanisme abordé en section 2.5). Les problèmes de sécurité liées à l'exécution d'applications malveillantes restent à considérer.

2.5 Java Web Start et JNLP

Le protocole JNLP permet de démarrer une application Java Web Start délivrée par le réseau. L'utilisation de ce protocole se base sur des fichiers JNLP (identifiés par une extension du même nom) qui contiennent différentes informations sur l'application à exécuter, comme par exemple :

- les versions compatibles du JRE ;
- l'URL de l'application et du fichier JNLP ;

8. Pour plus d'informations sur les limitations des appliquettes Java : https://docs.oracle.com/javase/tutorial/deployment/applet/security.html.

– le titre de l'application, son éditeur, sa description, etc. ;
– les systèmes d'exploitation et architectures pris en charge ;
– les permissions requises (pour demander toutes les permissions locales et réseau, ou demander l'exécution en bac à sable avec les seules permissions spécifiées par l'environnement du client) ;
– la liste des ressources utilisées par l'application.

Le processus de démarrage d'une application Java Web Start par son raccourci JNLP se résume en trois étapes :

– l'utilisateur démarre une application Java Web Start en exécutant un raccourci JNLP présent dans son arborescence de fichiers ou téléchargé avec son navigateur Web ;
– à l'aide du protocole JNLP, Java Web Start (qui fait partie du JRE) interroge le serveur Web spécifié dans le raccourci JNLP et vérifie si l'application est déjà stockée dans le cache local. Si ce n'est pas le cas ou si la version en cache n'est pas à jour, Java Web Start télécharge la dernière version de l'application et la met en cache. L'application est ensuite exécutée ;
– si, lors de son exécution, l'application nécessite des ressources qui ne sont pas présentes en cache ou dont les versions en cache ne sont pas à jour, Java Web Start télécharge ces ressources et les met en cache.

2.6 NPAPI et le module complémentaire Java

Le module complémentaire Java des navigateurs Google Chrome et Mozilla Firefox repose sur une ancienne interface de programmation appelée NPAPI, introduite avec le navigateur Netscape et n'apportant pas de fonction de sécurité convenable du fait de l'exécution des plugins avec le niveau de privilège de l'utilisateur. Une vulnérabilité affectant un plugin NPAPI permet alors de compromettre la session ou le système.

Microsoft Internet Explorer fait exception puisqu'il exécute les appliquettes Java via un contrôle ActiveX (ce navigateur étant le seul à prendre en charge les contrôles ActiveX). Toutefois, l'avenir d'Internet Explorer, et des contrôles ActiveX en général, reste incertain puisque ce navigateur historique risque d'être abandonné au profit de Microsoft Edge.

Le modulaire complémentaire Java NPAPI, et donc indirectement les appliquettes Java, ont une obsolescence annoncée qu'il convient d'anticiper tant par les développeurs d'applications Java que par les équipes d'administration des systèmes d'information :
– depuis septembre 2015 (Chrome version 45), les modules NPAPI ne sont plus supportés par Google Chrome ;
– la fin de support des modules NPAPI par Mozilla Firefox a été annoncée pour fin 2016 ;
– Microsoft Edge ne permet pas d'exécuter des appliquettes Java ;
– Microsoft Internet Explorer 11 reste l'unique navigateur qui sera en mesure d'exécuter des appliquettes Java après fin 2016.

R1 - Anticiper l'obsolescence annoncée des appliquettes Java

Il est recommandé d'anticiper l'obsolescence annoncée des appliquettes Java, en remplaçant ou en faisant évoluer les applications reposant sur cette technologie.

3 La phase d'inventaire

La phase d'inventaire est indispensable pour bien appréhender la démarche de sécurisation et aborder de manière optimale les différents cas de figure qui se présentent.

3.1 Recensement des applications Java et de leurs utilisateurs

Il est important d'identifier les applications Java, lourdes et légères, qui sont utilisées au sein du SI. La direction des systèmes d'information pourra ensuite définir une stratégie de sécurisation sur la base de son inventaire d'applications. Si un tel recensement n'existe pas, plusieurs méthodes de recherche peuvent être envisagées :

- en consultant d'une part l'outil d'inventaire ou de gestion de parc, s'il existe ;
- en interrogeant d'autre part le personnel qui procède aux installations ou au déploiement des modèles de postes utilisateurs, ainsi que les responsables des applications identifiées pour en connaître les utilisateurs légitimes ;
- en consultant les droits et groupes utilisateurs des annuaires et solutions de SSO, si l'accès à certaines applications est restreint ce niveau ;
- en cherchant des extensions `.jar`, `.class` ou `.jnlp` sur les serveurs d'application et espaces de stockage centralisés.

Les applications lourdes Java ne sont pas nécessairement considérées comme des applications installées, elle ne sont pas toujours remontées par les outils d'inventaire, scripts WMI, et méthodes équivalentes. La voie organisationnelle reste alors incontournable pour un recensement précis.

R2 - Recenser les applications Java

Recenser les applications Java, lourdes, légères et Java Web Start, ainsi que leurs utilisateurs légitimes.

3.2 Recensement des postes munis de JRE et des versions déployées

Une fois les applications Java recensées, la deuxième étape consiste à dresser la liste des postes sur lesquels des JRE sont présents. Si une telle liste n'existe pas au niveau de la DSI, des méthodes simples permettent de l'obtenir :

- à l'aide d'un outil d'inventaire ou de gestion de parc, s'il existe ;
- au moyen de scripts.

R3 - Recenser les installations de Java

Recenser les postes de travail sur lesquels des JRE sont installés, et leurs versions.

Des méthodes et exemples de scripts d'inventaire sont proposés en annexe A.

3.3 Pertinence de la présence de JRE

Sur la base du croisement des recensements précédemment effectués, il est à présent possible de repérer des anomalies :

– les postes utilisateurs sur lesquels un JRE est inutilement installé ;
– les postes utilisateurs sur lesquels un module complémentaire de JRE (NPAPI sous Mozilla Firefox ou ActiveX sous Microsoft Internet Explorer) est inutilement activé dans le navigateur.

R4 - Recenser les modules complémentaires Java inutiles

Repérer les postes utilisateurs sur lesquels des JRE ou des modules complémentaires Java sont inutilement présents.

4 La démarche de sécurisation

Pour réduire la surface d'attaque, il est primordial de limiter l'installation des JRE au périmètre des postes utilisateurs requérant cet environnement. Les anomalies recensées lors de la phase d'inventaire précédente devront faire l'objet de mesures de correction.

4.1 Interdiction des JRE inutiles

De manière générale, les JRE inutiles doivent être interdits. Suite à leur désinstallation, des mesures techniques et organisationnelles seront mises en place pour garantir l'absence de Java sur ces postes. Parmi ces mesures de sécurité, citons par exemple :
– le retrait des droits d'administration locaux pour les comptes utilisateurs, selon le principe de moindre privilège ;
– l'utilisation de fonctionnalités d'*Active Directory* telles que SRP (*Software Restriction Policies*, ou politiques de restriction logicielles) ou AppLocker ;
– l'utilisation de fonctionnalités de contrôle des applications installées, offertes par les principales suites de protection des postes de travail et serveurs ;
– la sensibilisation des utilisateurs ;

R5 - Désinstaller les JRE inutiles

Interdire les JRE lorsqu'ils sont inutiles, les désinstaller, et mettre en place des mesures de sécurité techniques et organisationnelles empêchant leur installation.

R6 - Définir des exceptions pour les postes utilisateurs ayant besoin du JRE

Si seuls quelques postes utilisateurs requièrent un JRE pour l'exécution d'applications bien spécifiques, il est recommandé d'interdire les JRE au niveau du SI et de mettre en place des exceptions techniques et organisationnelles pour les postes utilisateur concernés.

4.2 Désactivation des modules complémentaires Java et des Java Web Start inutiles

Dans le cas où seules des applications lourdes Java sont utilisées, il est fortement recommandé de mettre en place les mesures de sécurité interdisant l'utilisation du module complémentaire Java dans

les navigateurs ainsi que Java Web Start.

R7 - Filtrer au niveau des serveurs mandataires

Autant que possible, bloquer les appliquettes Java (extensions `.jar` et `.class`, types MIME `application\x-java-*`, etc.) et les fichiers JNLP au niveau des serveurs mandataires et autres équipements de filtrage en coupure du protocole HTTP.

Si aucun filtrage n'est opéré en coupure de la navigation sur Internet, il s'agit d'un point d'amélioration qui devrait être considéré au niveau de l'architecture du SI. Le guide de l'ANSSI sur la définition d'une architecture de passerelle sécurisée [9] aborde cet aspect.

R8 - Désactiver les modules complémentaires Java dans les navigateurs

Forcer la désactivation des modules complémentaires Java dans les navigateurs, par le biais de stratégies de sécurité adaptées à chacun des navigateurs utilisés.

Cette recommandation requiert de maîtriser les navigateurs déployés au sein du SI [10].

R9 - Désactiver Java dans les navigateurs

Autant que possible, désactiver complètement le contenu Java pour les navigateurs par configuration du JRE.

Cette option concerne à la fois les modules complémentaires et Java Web Start. Elle peut s'effectuer manuellement par le panneau de configuration Java, ou par utilisation des fichiers de configuration `deployment.properties` comme indiqué en annexe B.

R10 - Définir des exceptions pour les postes utilisateurs ayant besoin de Java dans un navigateur

Si seuls quelques utilisateurs accèdent à des applications Java légères, il est recommandé d'interdire les modules complémentaires Java et de mettre en place des exceptions techniques et organisationnelles pour les utilisateurs concernés.

9. Voir http://www.ssi.gouv.fr/passerelle-interconnexion.
10. L'ANSSI a publié des recommandations de sécurisations des navigateurs Mozilla Firefox, Google Chrome et Microsoft Internet Explorer disponibles sur son site Internet http://www.ssi.gouv.fr.

4.3 Remplacement des appliquettes Java

Dans le cas où le langage Java doit nécessairement être utilisé et qu'aucune solution de remplacement vers une autre technologie n'est envisageable à court terme, plusieurs axes de travail simples et rapides à mettre en œuvre peuvent être envisagés pour, au minimum, s'affranchir des problèmes d'obsolescence annoncée des appliquettes Java dans les navigateurs.

4.3.1 Ré-écriture d'une appliquette Java en applications Java Web Start

La solution la plus complète et préférable de migration est une ré-écriture de l'appliquette en une application Java Web Start autonome. Elle pourra ainsi bénéficier complètement des fonctionnalités de Java Web Start. Le travail principal de ré-écriture consiste à convertir la classe `applet` principale en une classe `main` d'application Java. Les méthodes `init` et `start` ne sont plus présentes et l'application doit alors s'initialiser en début de méthode `main` (les appels effectués en `init` et `start` doivent donc être déplacés en début de `main`).

R11 - Remplacer les appliquettes Java

Dès lors que des appliquettes Java doivent être utilisées et qu'aucune solution de remplacement vers une autre technologie (HTML5 entre autres) n'est envisageable à court terme, il est alors recommandé de ré-écrire ces appliquettes Java en applications Java Web Start. Cela permet de s'affranchir des modules complémentaires Java greffés aux navigateurs et d'anticiper l'obsolescence annoncée des appliquettes Java.

4.3.2 Migration simple des appliquettes Java en applications Java Web Start

Si une ré-écriture en application Java Web Start n'est pas envisageable, la technologie Java Web Start intègre un support des appliquettes. Il est ainsi possible d'exécuter une appliquette directement au sein de Java Web Start sans aucune recompilation de l'appliquette et sans nécessiter un navigateur (et, par conséquent, sans avoir à activer des modules complémentaires ou contrôles ActiveX au sein de ces derniers). Pour cela, il suffit de modifier les tags HTML d'une page Web appelant une appliquette pour fournir un fichier `JNLP` [11] à la place.

Ce fichier `JNLP` devra contenir, entre autres, les éléments suivants :

```
# ../.. Début de fichier JNLP incluant la version, le codebase, son lien href , les
# informations telles que le titre , etc.
# Ajouter la ressource SwingSet2.jar comme ceci :
<resources>
  <jar href="SwingSet2.jar"/>
</resources>
# Utiliser l'élément applet-desc et lui passer des arguments comme cela :
<applet-desc main-class='SwingSet2Applet' name='SwingSet' width='625' height='595'>
  <param name='param1' value='value1'/>
  <param name='param2' value='value2'/>
</applet-desc>
# ../.. Fin de fichier JNLP.
```

Au niveau d'une page Web, l'appliquette est ensuite remplacée par un simple lien de type :

```
<a href="MonApplication.jnlp"> Lancer l'application</a>
```

11. Voir le tutorial JNLP sur le site d'Oracle : https://docs.oracle.com/javase/tutorial/deployment/deploymentInDepth/jnlpFileSyntax.html.

L'application peut également être exécutée depuis un raccourci `MonApplication.jnlp` présent sur le poste de travail de l'utilisateur, ou bien encore avec une simple commande comme par exemple :

```
$ javaws http://hote:port/chemin/MonApplication.jnlp
```

Certaines considérations spécifiques sont toutefois à prendre en compte (cache, cookies, JSObject, etc.), le lecteur est donc invité à consulter l'article relatif à la migration d'appliquettes Java sur le site d'Oracle [12].

R11⁻ - Migrer les appliquettes Java

À défaut d'une ré-écriture en application Java Web Start, il reste toujours possible de simplement migrer les appliquettes sans nécessiter leur recompilation.
Cette opération permet de prendre en compte la fin du support des appliquettes Java dans les navigateurs mais, en revanche, cela ne permet pas d'anticiper l'obsolescence des appliquettes à partir de Java 9.

Attention de ne pas ré-écrire en Java Web Start une application qui s'apparenterait à un remplacement de module complémentaire et qui apporterait des problèmes de sécurité similaires. À titre d'exemple, développer en Java Web Start un serveur local en écoute qui communique avec le navigateur et le serveur de manière à garder l'interface graphique Web inchangée pour le client est une mauvaise pratique.

4.4 Configuration centralisée

Avant tout déploiement du JRE, il est fortement recommandé de commencer par définir une configuration centralisée qui sera appliquée aux différents postes utilisateur. Le fichier `deployment.properties` est prévu à cet effet. Il s'agit d'un fichier de configuration au format texte brut qui contient l'ensemble des paramètres de sécurité que l'on retrouve dans le panneau graphique de configuration Java accessible depuis le panneau de configuration Windows.

Ce fichier est par défaut présent localement sur les postes utilisateurs, dans le profil de chaque utilisateur. Il peut toutefois également s'appliquer au niveau système pour tous les utilisateurs. En plaçant ce fichier dans un espace centralisé accessible aux postes de travail (serveur Web ou partage CIFS par exemple), il est alors possible de définir une configuration centralisée qui aura pour avantages :

- d'être unique et donc homogène à l'échelle du système d'information, ou au contraire d'avoir plusieurs variantes spécifiques à certains groupes de postes de travail ;
- d'être appliquée instantanément sur les postes de travail après modification, pour ainsi pouvoir réagir rapidement en cas d'alertes de sécurité ;
- de pouvoir être sauvegardée par simple copie de fichier ;
- de définir précisément des listes d'applications autorisées, interdites, devant s'exécuter avec une version spécifique du JRE ou encore pouvant s'exécuter malgré un niveau de sécurité en deçà du niveau minimum configuré ;
- de permettre le verrouillage des paramètres de configuration, qui ne seront donc pas modifiables par les utilisateurs.

12. Voir http://docs.oracle.com/javase/8/docs/technotes/guides/deploy/applet_dev_guide.html#CIADJHDC.

Il s'agit donc d'une fonctionnalité importante du JRE. Sa bonne mise en œuvre est une mesure de sécurité qu'il convient de ne pas ignorer, tant son bénéfice est important dans la démarche de sécurisation des environnements d'exécution Java.

Sa mise en œuvre se fait en deux étapes :

- le dépôt du fichier `deployment.properties` dans un espace accessible aux postes de travail (serveur Web ou partage CIFS par exemple). Attention à bien prendre en compte les problèmes d'accessibilité de ce fichier pour les postes en situation de mobilité (en lecture seule pour tous les utilisateurs, et en écriture seulement localement sur le serveur) ;
- le dépôt d'un fichier `deployment.config` sur l'ensemble des postes de travail Windows, à l'emplacement `%windir%\Sun\Java\Deployment\` et qui indique l'URL du fichier `deployment.properties` centralisé.

Cette fonctionnalité, disponible depuis Java 5, est détaillée sur le site d'Oracle dans la documentation en ligne de Java 8 [13].

L'annexe B guide dans la mise en place d'une configuration centralisée des JRE par stratégies de groupe (GPO) dans un domaine *Active Directory*.

R12 - Définir une configuration sécurisée de Java verrouillée

Définir une configuration sécurisée, homogène, centralisée et verrouillée par le biais des propriétés de déploiement (fichiers `deployment.properties` et `deployment.config`).

R12$^-$ - Définir une configuration sécurisée de Java non verrouillée

Définir une configuration sécurisée, homogène et centralisée par le biais des propriétés de déploiement (fichiers `deployment.properties` et `deployment.config`).
Cependant, sans verrouillage de l'ensemble de la configuration, l'utilisateur sera en mesure de modifier les éléments de configuration du JRE non verrouillés par le biais du panneau de configuration (entre autres). Cela peut donc entraîner un affaiblissement de la sécurité de l'environnement d'exécution Java.

4.5 Télédéploiement du JRE

Lorsque le JRE est nécessaire à l'exécution de certaines applications Java publiées au sein du système d'information, se pose alors la question de son installation sur les postes utilisateurs concernés. Le JRE, au même titre que les autres logiciels, devrait être installé sur les postes de travail par télédéploiement. Celui-ci est une des pratiques fondamentales d'un système d'information contrôlé et maîtrisé. En effet, il permet de maîtriser les installations, d'homogénéiser les configurations, et de procéder aux mises à jour de manière réactive et efficace sur l'ensemble du parc.

Les différentes versions de Java se succédant à une fréquence élevée, il est d'autant plus important de pouvoir les déployer rapidement dès leur publication. Il est donc fortement recommandé de mettre en œuvre cette pratique, et d'associer son application à des procédures organisationnelles et techniques éprouvées et maîtrisées. *Active Directory* dispose de fonctionnalités de télédéploiement mais des plateformes de déploiement sont également souvent utilisées. Dans cette optique, Oracle fournit les paquets d'installation du JRE (32 et 64 bits) dans une version installable hors-ligne (c'est-à-dire ne nécessitant pas de connexion Internet lors de l'installation) qu'il est possible de récupérer :

13. Voir https://docs.oracle.com/javase/8/docs/technotes/guides/deploy/properties.html.

– sous forme d'exécutable (extension `.exe`), auquel il est possible d'indiquer des options d'installation par l'intermédiaire d'arguments de ligne de commande ;
– sous forme de MSI (extension `.msi`) dont les options d'installation peuvent être spécifiées dans un fichier de transformation (extension `.mst`).

Les options d'installation vont permettre de maîtriser la manière dont le JRE est déployé, configuré, et mis à jour. Elles vont par exemple permettre de spécifier si les modules complémentaires de Microsoft Internet Explorer ou de Mozilla Firefox doivent être activés. Ces paramètres sont toutefois inutiles lorsqu'une configuration centralisée par fichier `deployment.properties` est mise en œuvre. Le télédéploiement peut alors se faire de plusieurs manières, les principales et les plus communément utilisées étant :
– par GPO (stratégies de groupe pour la gestion centralisée) dans un domaine Windows ;
– par l'utilisation de *Microsoft System Center Configuration Manager* ou de tout autre produit tiers prévu à cet effet.

L'annexe C de ce document contient des informations sur le télédéploiement du JRE par GPO dans un domaine *Active Directory*.

R13 - Restreindre le périmètre de déploiement du JRE

Déployer le JRE par télédéploiement, sur le périmètre le plus restreint possible correspondant aux postes utilisateur légitimes référencés.

À défaut de pouvoir désinstaller les anciennes versions du JRE sur les postes de travail, il est important d'en déployer la dernière version de manière à ce qu'elle prenne en charge et applique la configuration centralisée précédemment définie (voir section 4.4). Cette configuration centralisée ne permet en revanche pas de désactiver explicitement les versions non gérées en télédéploiement.

4.6 Gestion des mises à jour

Une gestion des mises à jour réactive et efficace est cruciale pour le maintien en condition de sécurité du SI. Cela implique de se tenir informé des alertes de sécurité, et de rapidement déployer les nouvelles versions du JRE.

R14 - Faire de la veille

Se tenir informé des alertes de sécurité relatives à Java, ainsi que celles des autres applications recensées dans le système d'information de l'entreprise.

Le site du CERT-FR propose par exemple un flux RSS complet d'avis, alertes, et bulletins d'information [14].

R15 - Maintenir les installations logicielles à jour

Les applications installées sur les postes de travail doivent être mises à jour dans les plus brefs délais dès lors qu'une version plus sécurisée est disponible.

5 Pour aller plus loin

5.1 Cas particulier des systèmes d'exploitation Microsoft Windows 64 bits

Sur les systèmes d'exploitation Microsoft Windows 64 bits cohabitent les versions 32 et 64 bits des JRE. Les versions 32 bits sont installées par défaut dans `C:\Program Files (x86)\` et les versions 64 bits dans `C:\Program Files\`.

Les versions 64 bits du JRE copient leurs principaux exécutables (`java.exe, javaw.exe, et javaws.exe`) dans le répertoire `%System32%`. À cet emplacement se trouve toujours la version la plus récente de ces exécutables. De fait, dès lors qu'une version 64 bits du JRE est installée, elle est systématiquement utilisée par défaut lors de l'exécution d'une application lourde. De même, la version utilisée sera toujours la plus récente, y compris si un JRE7 est installé après un JRE8. Il est possible de voir la version utilisée en tapant `java -version` en ligne de commandes.

En revanche, il est important de noter que Microsoft Internet Explorer est par défaut utilisé dans sa version 32 bits lorsqu'il est exécuté depuis les raccourcis présents par défaut sur le bureau ou dans la barre de lancement rapide, c'est à dire dans la grande majorité des cas d'utilisation. La version 64 bits dénommée *Internet Explorer (64 bits)*, est quant à elle accessible entre autres depuis le menu Démarrer, mais reste peu souvent utilisée. Internet Explorer dans sa version 32 bits n'utilisera qu'une version 32 bits du JRE, tandis que la version 64 bits d'Internet Explorer n'utilisera que la version 64 bits du JRE. Il est fréquent de voir cohabiter des versions 32 et 64 bits du JRE sur un même poste utilisateur, et de versions différentes.

Par ailleurs, la mise à jour du JRE 64 bits ne corrige en rien les vulnérabilités du JRE 32 bits installé, et la navigation Web via Internet Explorer 32 bits restera vulnérable face à une appliquette Java malicieuse, ce qui peut s'avérer être trompeur pour l'utilisateur ou l'administrateur système.

R16 - Mettre à jour le JRE 32 bits sur les systèmes 64 bits

Sur les systèmes 64 bits, il est nécessaire de mettre à jour non seulement la version 64 bits du JRE, mais également la version 32 bits généralement utilisée par Microsoft Internet Explorer.

Pour télédéployer uniquement la version 64 bits du JRE sur les postes utilisateurs 64 bits, il est possible d'utiliser la fonctionnalité de filtre WMI des stratégies de groupes (dans l'espace de noms `root\CIMV2` avec la requête `Select * from Win32_Processor where AddressWidth ='64'`).

14. Le flux RSS du CERT-FR est disponible à l'adresse suivante http://www.cert.ssi.gouv.fr/site/cert-fr.rss.

5.2 Fichiers `.jar` et `.jnlp`

R17 - Considérer les extensions JAR et JNLP au même titre que des exécutables

Lorsque des JRE sont déployés au sein du SI, il est important de considérer les fichiers d'extension `.jar` et `.jnlp` comme des exécutables à part entière, représentant une menace similaire. Ces fichiers peuvent en effet être exécutés par simple double-clic et ainsi exploiter les vulnérabilités du JRE pour compromettre des postes de travail. Les règles qui s'appliquent aux extensions `.exe` devraient donc également s'appliquer aux `.jar` et `.jnlp`, comme par exemple sur les serveurs mandataires ou de filtrage des courriels.

5.3 Navigateurs multiples

Si les ressources de l'entité le permettent et si les contraintes le justifient (en attendant par exemple un remplacement des appliquettes par du Java Web Start), une bonne stratégie consiste à déployer, maîtriser, et maintenir deux navigateurs Web sur les postes utilisateurs :

- un premier navigateur dédié à la navigation sur des sites de confiance en intranet muni des modules complémentaires Java ;
- un deuxième navigateur dédié à la navigation sur Internet, dépourvu des modules complémentaires.

Cette démarche doit s'accompagner de toute mesure de sécurité complémentaire assurant d'une part la séparation des usages et d'autre part la sécurité de la navigation sur Internet. Entre autres :

- seule l'utilisation de serveurs mandataires doit permettre de naviguer sur Internet, et ces derniers devraient encore une fois bloquer les appliquettes Java ;
- le `User-Agent` du deuxième navigateur doit être le seul autorisé à sortir sur Internet au niveau des serveurs mandataires ;
- par stratégie de sécurité sur le pare-feu local des postes de travail, le premier navigateur ne devrait pouvoir se connecter qu'aux plages d'adresses IP internes, tandis que le deuxième ne devrait pouvoir se connecter qu'au serveur mandataire ;
- les utilisateurs doivent être sensibilisés et bien comprendre les raisons de cette séparation des usages ;
- en cas de vulnérabilité, des procédures éprouvées doivent permettre d'appliquer dans des délais très brefs les correctifs de sécurité au navigateur web utilisé pour Internet voire de le remplacer.

R18 - Maintenir et maîtriser deux navigateurs

Si les ressources de l'entité le permettent et si les contraintes le justifient, il peut être bénéfique de déployer, maintenir et bien maîtriser deux navigateurs Web sur les postes utilisateurs. Cela permet de séparer leurs usages et ainsi d'éviter d'avoir un navigateur trop permissif pour l'accès à du contenu potentiellement malveillant.

Cette recommandation ne doit être appliquée que si l'on est en mesure d'intégrer deux navigateurs dans le SI dans de bonnes conditions.
Dans la majorité des situations, il est préférable de disposer d'un seul navigateur complètement maîtrisé, plutôt que de deux navigateurs peu ou non maîtrisés.

5.4 Intégrer le MCO et le MCS dans les contrats

Il est fréquent que des applications Java soient développées dans le cadre de marchés ou de projets où le maintien en condition de sécurité n'est pas prévu. Dans de nombreux cas, ces applications ne fonctionnent ou ne sont maintenues que pour une seule version majeure voire pour une seule version mineure du JRE. Ces contraintes génèrent des coûts de maintenance pour le parc des postes utilisateurs concernés sur lequel il est nécessaire de maintenir plusieurs versions du JRE mais elles gènerent également des problèmes de sécurité en obligeant à maintenir installées des versions vulnérables. Pour éviter cette situation, il est donc nécessaire de prévoir une clause portant sur le MCO et le MCS de l'application dans le cadre du marché.

R19 - Intégrer les MCO et MCS du JRE dans les projets

> Bien intégrer la sécurité dans les projets, de manière à ce que la mise à jour du JRE soit garantie par les contrats de MCO (maintien en conditions opérationnelles) et de MCS (maintien en condition de sécurité). Les applications doivent au plus vite être rendues compatibles avec les nouvelles versions majeures lorsqu'elles apportent des évolutions entraînant des incompatibilités.

A Scripts d'inventaire des JRE

Un outil de gestion de parc, et en l'occurrence d'inventaire logiciel, permet d'avoir rapidement connaissance des applications installées sur les différents postes de travail. Il est fortement conseillé de disposer d'un tel outil. L'utilisation de scripts pour recencer les JRE installés au sein du système d'information ne devrait être envisagée que comme solution palliative. Les scripts proposés dans cette annexe sont donnés à titre illustratif et de manière à servir de piste de réflexion, ils doivent être adaptés et utilisés avec précaution.

A.1 Script d'inventaire en PowerShell

Un script PowerShell pourrait être utilisé pour automatiser le recensement des JRE installés au sein d'un domaine *Active Directory*.

L'exécution de ce script requiert PowerShell en version 1 ou ultérieure, ainsi que des privilèges d'administration. Les postes de travail doivent également répondre aux requêtes WMI (le service d'appel de procédure distante doit être démarré, et RPC doit être autorisé au niveau de leur pare-feu local). Ce script n'interroge qu'un poste de travail à la fois, son exécution peut donc prendre plusieurs heures sur un parc volumineux. Il remonte par ailleurs les versions des JRE installés mais ne donne aucune indication sur la présence ou non de modules complémentaires Java activés dans les navigateurs.

```
# initialisation
import-module activedirectory
$compteur = 0
$sortie = "./inventairejava" + (Get-Date -format "yyyyMMdd").ToString() + ".csv"
"ordinateur;produit;version" > $sortie

# liste des ordinateurs du domaine
$ordinateurs = Get-AdComputer -Filter * | select Name

# pour chaque ordinateur, liste des produits Java par WMI
foreach($ordinateur in $ordinateurs)
{
  # barre de progression
  start-sleep 1
  write-progress -activity $("inventaire␣en␣cours␣(␣" `
    + $ordinateurs.count + "␣machines␣à␣analyser␣)") -status "%␣réalisé␣:"`
    -percentcomplete $($compteur*100/$ordinateurs.count)
  # Les noms de JRE ne sont pas standards d'une version à l'autre, on cherche
  # donc "%Java%". Les JRE, JDK, et Auto Updater seront remontés par le script.
  foreach ($Element in $(Get-WmiObject -errorAction SilentlyContinue `
    -errorVariable Erreur -computer $ordinateur.name -query `
    "select␣name,␣version␣from␣win32_product␣where␣name␣LIKE␣'%Java%'"))
  {
    if($Element)
    {
      $($ordinateur.name + ";" + $Element.name + ";" + $Element.version) >> $sortie
    }
  }
  $compteur++
  if($Erreur)
  {
    $($ordinateur.name + ";injoignable(" + $Erreur[0].Exception.message + ");") `
      >> $sortie
    $Erreur.clear()
  }
}
"L'inventaire␣est␣terminé,␣il␣a␣été␣enregistré␣dans␣le␣fichier␣" + $sortie
```

A.2 Script d'inventaire sur page Web

Une deuxième approche consiste à intégrer du code à un site Web tel que l'intranet, fréquemment visité par les utilisateurs. Cette méthode a pour avantage de permettre d'inventorier plus spécifiquement les versions de modules complémentaires Java des postes utilisateurs qui s'y connectent. Si cette démarche est adoptée, sa mise en œuvre doit être coordonnée par la DSI.

Oracle fournit un script Javascript [15], qui pourrait être utilisé au sein d'une page HTML spécialement conçue. Il s'agit par contre d'un script qui s'exécute côté client, nécessitant donc une redirection vers une page exécutant du script côté serveur de manière à pouvoir stocker l'information en bases de données. Il y a alors un traitement de l'information en deux temps.

Si les `iFrames` HTML sont supportées par les navigateurs utilisés, la page principale du site intranet pourrait par exemple être modifiée de la manière suivante :

```
<!doctype html>
<html><head><title>Un Titre</title></head>
<body>
<h1>Ici votre page Web habituelle</h1>
<!-- Ici un iframe invisible à ajouter,
dont la source est la page contenant le script d'inventaire -->
<iframe src="inventairejava.html" width=1 height=1 style="display:␣none;"></iframe>
</body>
</html>
```

L'avantage d'une `iFrame` est son invisibilité, permettant une intégration transparente pour les utilisateurs. Si toutefois les `iFrames` ne pouvaient pas être utilisées, un remplacement complet de la page principale du site intranet pourrait également être envisagé.

La page `inventairejava.html` de l'`iFrame`, positionnée dans le même répertoire que la page ci-dessus et que le script `DeployJava.js`, contient le script Java exécuté côté client :

```
<!doctype html>
<html><head><title>Un Titre</title>
<script type="text/javascript" src="./deployJava.js"></script>
<!--
- Ici la redirection vers une page à créer dans le language de votre choix
et appelée inventairejre.php dans cet exemple.
- Cette page récupèrera en arguments l'ip du client et la version de son module
complémentaire Java si toutefois il est activé.
- La page à creer doit ajouter ces informations à une base de données de manière
à pouvoir exploiter l'information par la suite.
-->
<script type="text/javascript">
  document.location.href="inventairejava.php?ordinateur=" + window.location.host
  + "&java=" + deployJava.getJREs();
</script>
</head>
<body>
<script>
  document.write(navigator.userAgent);
</script>
</body>
</html>
```

Pour finir, une dernière page (`inventairejava.php` dans cet exemple) contenant un script exécuté côté serveur est nécessaire. Elle aura pour rôle de stocker en base de données les informations qui lui seront communiquées par la page précédente. Cette page doit être codée avec un langage supporté par le serveur Web utilisé (ASP ou PHP par exemple). Une telle page en langage PHP ressemblerait par exemple à cela :

15. Script disponible à l'adresse http://java.com/js/deployJava.js.

```
<!doctype html>
<html>
<head><title>Un titre</title></head>
<body>
<?php
// fonctions de sécurité
// Cet exemple de script se base sur l'utilisation d'un SGBD
// MySQL. Ces fonctions de sécurité spécifiques à MySQL peuvent être
// remplacées par des PDO (http://php.net/manual/fr/book.pdo.php), plus
// génériques et utilisables avec d'autres SGBD. Il est dans tous
// les cas important à bien sécuriser le script PHP de manière à ce
// qu'il ne présente pas de vulnérabilités dans le traitement des
// données en entrée.
function securise($string){
  $string = mysql_real_escape_string($string);
  $string = addcslashes($string, '%_');
  return $string;
}
try{
  // Connexion à la BDD
  $bdd = mysqli_connect("127.0.0.1", "utilisateur", "motdepasse", "ma_base");
  // On contrôle la cohérence des données passées en argument
  $ordinateur = securise($_GET['ordinateur']);
  $java = securise($_GET['java']);
  // On ajoute une entrée dans la base, ou on UPDATE la version de Java
  $requete = "INSERT␣INTO␣ma_table(ordinateur,␣java)
␣␣VALUES('$ordinateur',␣'$java')␣ON␣DUPLICATE␣KEY␣UPDATE␣java='$java'";
  mysqli_query($bdd,$requete) or exit(mysql_error());
}
catch(Exception $e){
  die('Erreur␣:␣'.$e->getMessage());
}
?>
</body>
</html>
```

Idéalement et si les ressources le permettent, cette méthode pourrait être améliorée de manière à remonter, sous forme d'alerte, l'apparition de modules complémentaires Java sur de nouveaux postes de travail.

B Configuration centralisée par fichiers

B.1 Création du fichier `deployment.properties`

Le fichier `deployment.properties` ci-dessous est un exemple de configuration, il ne doit en aucun cas être transposé en l'état sans en avoir préalablement mesuré les impacts. Les différents paramètres doivent être adaptés à chaque système d'information en fonction des besoins et du contexte. La configuration doit également être testée avant tout déploiement.

```
# fichier "deployment.properties".
# Il s'agit d'un exemple de configuration sécurisée
# Ces différents paramètres doivent être adaptés à chaque SI en fonction des besoins.
# Le fichier de configuration doit idéalement être testé avant tout déploiement.
# La présente configuration ne doit en aucun cas être transposée en l'état sans en avoir
# préalablement mesuré les impacts.
# Les propriétés ".locked" verrouillent la propriété du même nom de manière à ce qu'un
# utilisateur ne puisse pas les modifier.
# ------------------------------------------------------------------------------
# Paramétrage du proxy. Si une configuration spécifique doit être utilisée, se
# référer à la documentation officielle Java. Si, du fait du contexte, le module
# complémentaire Java ne peut pas être désactivé dans le navigateur, il peut par
# exemple être intéressant d'utiliser un proxy spécifique à Java et qui n'autorise
# la connexion qu'à une liste de sites de confiance bien définie.
# La valeur suivante indique d'utiliser le proxy du navigateur Web.
deployment.proxy.type=3
deployment.proxy.type.locked
# Ne pas associer des extensions de fichiers à java (sauf si vous désirez utiliser JNLP
# auquel cas utilisez par exemple la valeur 1).
deployment.javaws.associations=0
deployment.javaws.associations.locked
# Désactiver toute utilisation de modules complémentaires java et de Java Web Start.
# Mettre à true si Java Web Start et JNLP doivent être utilisés.
deployment.webjava.enabled=false
deployment.webjava.enabled.locked
# Niveau de sécurité très élevé (les appliquettes non signées ne seront pas exécutées).
# Le paramètre par défaut est HIGH et doit être changé si des problèmes de compatibilité
# se posent. En niveau HIGH, les appliquettes signées avec un certificat expiré ou dont
# la révocation ne peu pas être vérifiée sont autorisées.
deployment.security.level=VERY_HIGH
deployment.security.level.locked
# Forcer la vérification de révocation des certificats auprès de leur CRL. Si certains
# certificats n'ont pas de CRL opérationnelle, ce paramètre doit être positionne à false.
# (l'url de vérification peut être renseigné via la propriété
# deployment.security.validation.crl.url).
deployment.security.validation.crl=true
deployment.security.validation.crl.locked
# Si OCSP est utilisé (l'url de vérification peut être renseigné via la propriété
# deployment.security.validation.ocsp.url).:
deployment.security.validation.ocsp=true
deployment.security.validation.ocsp.locked
# Vérifier le statut de révocation de tous les certificats de la chaîne
deployment.security.revocation.check=ALL_CERTIFICATES
deployment.security.revocation.check.locked
# Choix manuel du certificat client même lorsqu'il y en à qu'un
deployment.security.clientauth.keystore.auto=false
deployment.security.clientauth.keystore.auto.locked
# Désactiver la possibilité pour l'utilisateur d'autoriser l'exécution d'appliquettes
# auto-signées. On considère que les appliquettes sont signées par une IGC de confiance.
deployment.security.askgrantdialog.notinca=false
deployment.security.askgrantdialog.notinca.locked
# L'utilisateur n'est pas autorisé à octroyer des droits d'accès
# élevés aux applications signées. Ce paramètre doit être mis à true si certaines
# applications signées utilisées nécessitent des droits d'accès élevés pour des cas
# particuliers.
deployment.security.askgrantdialog.show=false
deployment.security.askgrantdialog.show.locked
# Afficher le certificat du site serveur même s'il s'agit d'un certificat valide
deployment.security.https.warning.show=false
deployment.security.https.warning.show.locked
```

```
# Activer la vérification de révocation de liste noire de signatures de fichiers .jar
# (liste consultée par le plugin Java ainsi que par Java Web Start).
deployment.security.blacklist.check=true
deployment.security.blacklist.check.locked
# Afficher les bannières d'avertissement de la sandbox.
deployment.security.sandbox.awtwarningwindow=true
deployment.security.sandbox.awtwarningwindow.locked
# Avertir en cas de non concordance du certificat et du nom d'hôte.
deployment.security.jsse.hostmismatch.warning=true
deployment.security.jsse.hostmismatch.warning.locked
# Bloquer le code auto-signé
deployment.security.sandbox.selfsigned=NEVER
deployment.security.sandbox.selfsigned.locked
# Afficher les bannière d'avertissement de la sandbox pour permettre à
# l'utilisateur d'accepter des demandes d'autorisations JNLP.
deployment.security.sandbox.jnlp.enhanced=false
deployment.security.sandbox.jnlp.enhanced.locked
# Lorsqu'une application contient du code mixte, bloquer les composants non signés
# et sans avertissement. La valeur par défaut ENABLE peut être restaurée si des
# appliquettes utilisant du code mixte doivent être utilisées, mais il est préférable
# de chercher à faire évoluer les appliquettes en question lorsque c'est possible.
# Éviter d'utiliser les valeurs HIDE_RUN et DISABLE qui exécutent les applications
# contenant du code mixte sans avertissement, ou ne vérifient pas la présence de code
# mixte.
deployment.security.mixcode=HIDE_CANCEL
deployment.security.mixcode.locked
# Activer TLS et interdire SSL 3 (sauf en cas de contraintes fortes)
deployment.security.SSLv3=false
deployment.security.SSLv3.locked
deployment.security.TLSv1.2=true
deployment.security.TLSv1.2.locked
deployment.security.TLSv1.1=true
deployment.security.TLSv1.1.locked
deployment.security.TLSv1=true
deployment.security.TLSv1.locked
# Utiliser les mécanismes de sandboxing natifs de l'OS Windows (option apparue avec
# le JRE 8 update 51 et pas encore documentée)
deployment.security.use/native.sandbox=true
deployment.security.use/native.sandbox.locked
# Ne pas autoriser java à créer des raccourcis (sauf si vous utilisez JNLP)
deployment.javaws.shortcut=NEVER
deployment.javaws.shortcut.locked
# Minimiser la console java lors de son exécution.
deployment.console.startup.mode=DISABLE
deployment.console.startup.mode.locked
# Traces pour debogage éventuel.
deployment.trace=true
deployment.trace.locked
# Journalisation des erreurs.
deployment.log=true
deployment.log.locked
# Pas de téléchargement automatique du JRE, il sera télédéployé de manière centralisée.
deployment.javaws.autodownload=NEVER
deployment.javaws.autodownload.locked
# Pas d'icône dans la barre d'état système.
deployment.system.tray.icon=false
deployment.system.tray.icon.locked
# Ne pas afficher les exceptions du cycle de vie des appliquettes.
deployment.javapi.lifecycle.exception=false
deployment.javapi.lifecycle.exception.locked
# ------------------------------------------------
# Règles relatives aux magasins de certificats
# Attention, il est important que les magasins au niveau système ne soient accessibles
# par les utilisateurs qu'en lecture seule. L'utilisation d'un chemin de type
# $SYSTEM_HOME\Sun\Java\Deployment\magasin est recommandé.
# ------------------------------------------------
# Emplacement du magasin des AC racines de confiance au niveau système
deployment.system.security.cacerts=MON_CHEMIN\trustedcacerts
# Emplacement du magasin des AC racines de confiance au niveau système pour les
# Java Secure Socket Extensions
deployment.system.security.jssecacerts=MON_CHEMIN\trustedjssecacerts
```

```
# Emplacement du magasin de certificats de confiance au niveau système
deployment.system.security.trusted.certs=MON_CHEMIN\trustedcerts
# Emplacement du magasin de certificats de confiance au niveau système pour les
# Java Secure Socket Extensions
deployment.system.security.trusted.jssecerts=MON_CHEMIN\trustedjssecerts
#  ----------------------------------------
# Listes d'exception
# Dans cet exemple, le fichier exception.sites est stocké localement sur les postes
# et on considère que le fichier a été poussé par GPP. Ce fichier peut tout aussi
# bien être placé sur un hébergement Web et pointé par un URL HTTP ou HTTPS.
# Ce fichier contient une liste d'url d'applications Java autorisées à s'exécuter
# bien qu'elles seraient bloquées par les vérifications de sécurité configurées (à
# cause de certificats expirés ou non valides par exemple).
# Pour les appliquettes ou les ressources d'applications l'url doit être utilisé.
# Pour les applications Java Web Start, l'url du JNLP doit être utilisé.
# Le fichier exception.sites contient un url par ligne. Les URL supportent les
# protocoles FILE, HTTP et HTTPS. Il est important que ce fichier ne soit accessible
# qu'en lecture seule par les utilisateurs.
deployment.user.security.exception.sites=$SYSTEM_HOME\Sun\Java\Deployment\exception.sites
deployment.user.security.exception.sites.locked
```

Plutôt qu'une configuration laxiste en matière de sécurité, il est préférable d'opter pour une configuration durcie complétée par des exceptions de sécurité [16] (comme expliqué dans le fichier `deployment.properties` ci-dessus) pour les seules applications Java de confiance qui nécessitent une configuration moins stricte (application interne non signée par exemple).

Pour aller plus loin, il est également possible de configurer un *Deployment Rule Set* [17] qui liste précisément les applications Java autorisées et interdites (par leur certificat de signature et leur URL par exemple) et leur associe des actions correspondantes (bloquer, autoriser, forcer une version spécifique du JRE pour l'exécution, etc.).

Pour une liste exhaustive des propriétés de configuration prises en charge, se référer à la documentation de déploiement sur le site d'Oracle [18].

Le verrouillage de l'ensemble des paramètres permet de garantir qu'aucune modification de configuration ne sera possible par les utilisateurs. Le résultat d'une telle configuration se traduit par un panneau de configuration Java intégralement grisé sur les postes de travail comme l'illustre la figure 1.

Le fichier `deployment.properties` doit ensuite être déposé dans un espace accessible aux aux postes utilisateurs, comme par exemple un serveur Web ou un partage de fichiers CIFS. Plusieurs fichiers de configuration peuvent être déposés de manière à définir des configurations spécifiques à certains groupes de postes de travail.

B.2 Création et télédéploiement du fichier `deployment.config`

Le fichier `deployment.config` renseigne deux paramètres :
- l'emplacement du fichier `deployment.properties` créé précédemment ;
- le caractère obligatoire ou facultatif de lecture du fichier `deployment.properties` avant toute exécution de code Java.

Il doit être déposé sur tous les postes de travail utilisateurs sur lesquels va être déployé le JRE, à l'emplacement `%windir%\Sun\Java\Deployment\`. Il est important d'utiliser la variable d'environnement `%windir%` pour éviter d'indiquer un chemin en dur tel que `C:\Windows\` qui risque

16. Pour plus d'informations sur ces exceptions : https://docs.oracle.com/javase/8/docs/technotes/guides/deploy/exception_site_list.html.
17. Pour plus d'informations : https://docs.oracle.com/javase/8/docs/technotes/guides/deploy/deployment_rules.html.
18. Voir https://docs.oracle.com/javase/8/docs/technotes/guides/deploy/properties.html.

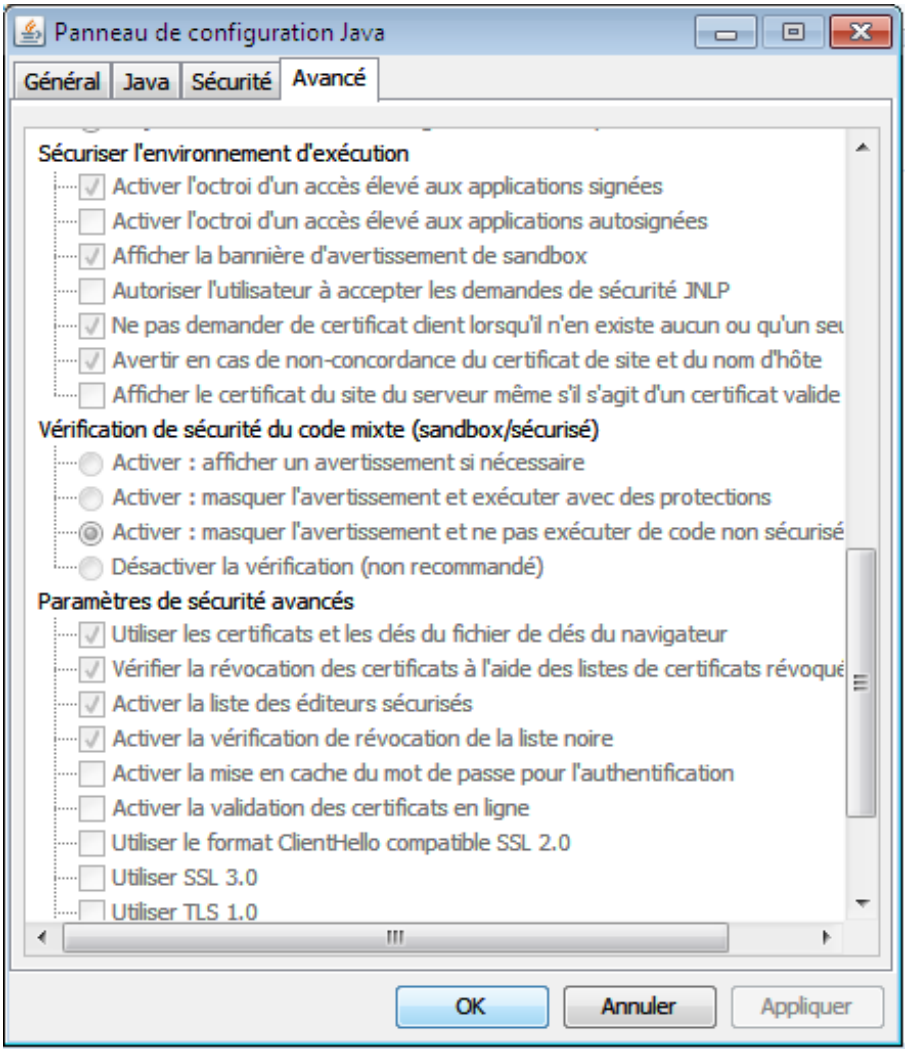

FIGURE 1 – Panneau de configuration Java verrouillé

de ne pas exister sur l'ensemble des postes de travail. Le fichier `deployment.config` suivant illustre l'utilisation de ces deux paramètres :

```
# Le premier paramètre indique l'url du fichier deployment.properties.
# Sur serveur web :
# deployment.system.config http://serveurweb/chemin/java/deployment.properties
# Sur partage de fichiers :
# deployment.system.config file:\\\\serveurweb\\partage\\java\\deployment.properties
# Attention à bien doubler les antislash.
deployment.system.config file:\\\\serveur\\partage\\deployment.properties
# Le deuxième paramètre rend obligatoire ou non l'accès au fichier
# deployment.properties. S'il est obligatoire et que le serveur est
# injoignable, aucun code ne sera exécuté. Il s'agit du paramètre recommandé, par
# contre il est important de prendre en compte la problématique d'utilisation de
# Java en situation de mobilité.
deployment.system.config.mandatory=true
```

Une simple stratégie de groupe permet dans un domaine *Active Directory* de télédéployer le fichier en question. Au niveau de la console de gestion des stratégies de groupe du domaine, cela se fait en créant et modifiant une nouvelle GPO, comme illustré par la figure 2.

À défaut de pouvoir centraliser le fichier `deployment.properties`, ou lorsque le contexte d'usage ne s'y prête pas, il peut également être envisagé de télédéployer ce fichier localement sur les postes de

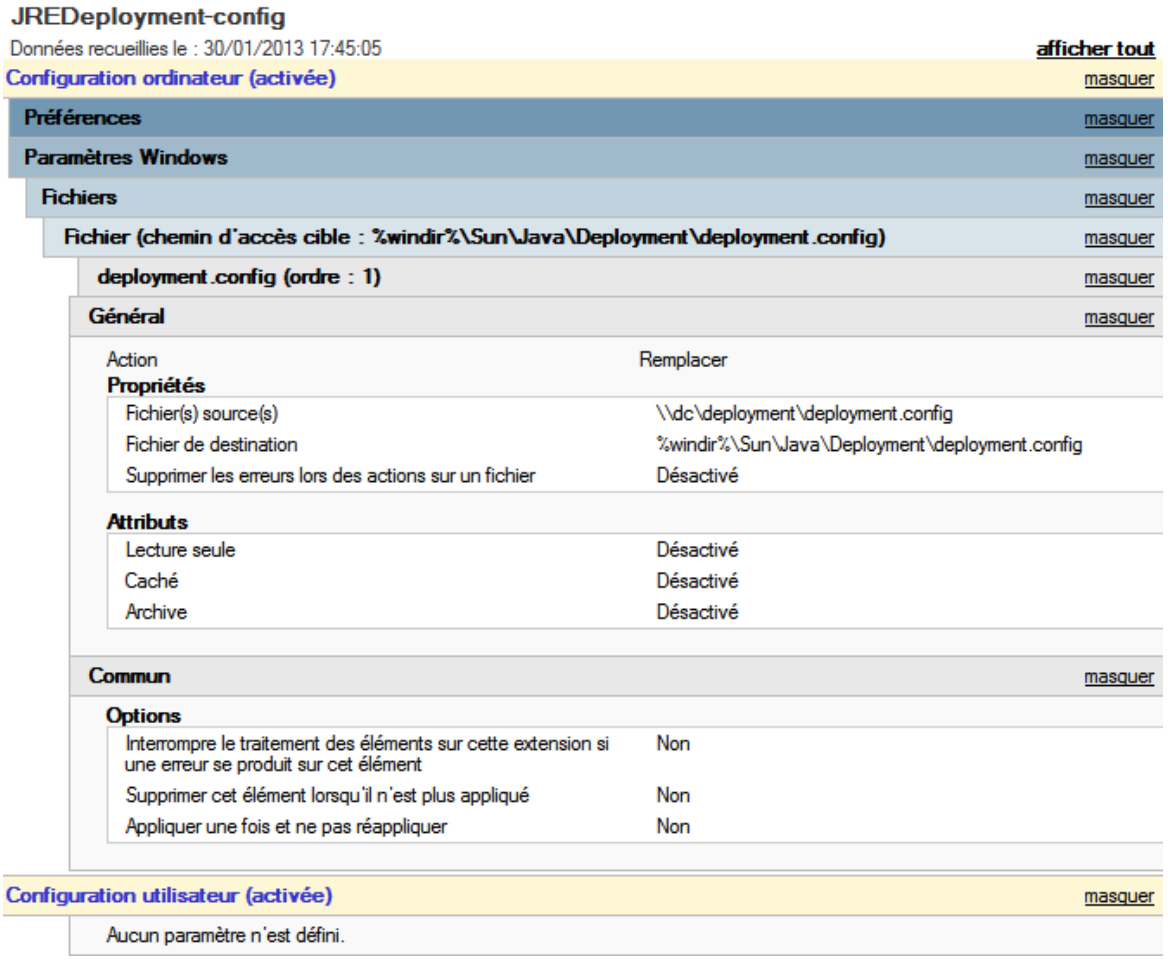

FIGURE 2 – GPO de déploiement du fichier deployment.config

travail. Le fichier `deployment.config` indique alors l'URL du fichier `deployment.properties` local qui peut être déployé par la même GPO.

C Télédéploiement du JRE 8 par GPO

Cette annexe présente de manière synthétique une méthode de télédeploiement reposant sur *Active Directory*, et illustre les quelques particularités du JRE à prendre en compte. Le lecteur est invité à prendre connaissance du guide complet d'installation du JRE 8 publié par Oracle [19].

C.1 Téléchargement du JRE

La dernière version du JRE est disponible sur le site Web de Java [20]. La version à récupérer est celle prévue pour une installation hors ligne, cette dernière étant proposée sous forme d'exécutable. Elle existe en 32 et en 64 bits (se référer à la section 5.1 qui traite ce sujet). L'exécutable téléchargé est signé numériquement afin de garantir son intégrité et sa provenance. La signature est issue d'un certificat de l'éditeur *Oracle*, rattaché à une autorité de confiance *Verisign* déjà présente dans le système d'exploitation. Une fois l'installeur téléchargé, il est recommandé de vérifier la validité de sa signature et la cohérence de la chaîne de certification avant de le déployer dans un système d'information. Cette vérification s'effectue dans l'onglet *Signature* des propriétés de l'exécutable.

Pour obtenir la version MSI du programme d'installation, il est possible d'exécuter le fichier précédemment téléchargé, d'annuler l'installation, puis de récupérer le fichier `.msi` qui a été décompressé à l'emplacement `\%userprofile%\AppData\LocalLow\Oracle\Java\`.

Attention toutefois, Oracle indique que l'extraction du MSI à partir du programme d'installation est une pratique non prise en charge, et Oracle ne garantit pas que les futures mises à jour de Java permettront toujours l'extraction du fichier MSI [21]. Depuis la sortie de Java SE 8 Update 20, les entités disposant de licences Java SE Advanced ou Java SE Suite ont accès à une version commerciale du JRE dont le programme d'installation est distribué au format MSI.

C.2 Déploiement par MSI

Pour un déploiement par fichier MSI, il est nécessaire d'avoir recours à un outil d'édition des MSI (comme ORCA par exemple, fourni gratuitement dans le SDK de Microsoft Windows). L'édition de MSI consiste à créer un fichier de transformation (fichier `.MST`) qui une fois joint au MSI indiquera des options d'installation spécifiques.

Pour créer un fichier de transformation avec ORCA, procéder comme suit :
- lancer ORCA ;
- ouvrir le fichier MSI d'installation du JRE (`jre1.8.0_92.msi` pour la version 8 Update 92) ;
- créer un nouveau fichier de transformation ;
- ouvrir la table `Property` (des propriétés d'autres tables peuvent également être modifiées pour répondre à des besoins spécifiques, mais dans la grande majorité des cas seule la table `Property` sera utile) ;
- modifier les propriétés en fonction des options d'installation désirées ;
- générer le fichier de transformation.

Les options d'installation suivantes peuvent par exemple être utilisées :
- pour désactiver les mises à jour automatiques, positionner `AUTOUPDATECHECK`, `JU` (ligne à créer) et `JAVAUPDATE` à la valeur 0. Les mises à jour devraient en effet être réalisées par de nouveaux

19. Voir http://docs.oracle.com/javase/8/docs/technotes/guides/install/windows_jre_install.html.
20. Voir http://www.java.com/fr/.
21. Voir les informations connexes en fin d'article dédié au déploiement de Java par MSI à l'adresse https://www.java.com/fr/download/help/msi_install.xml.

télédéploiements dès lors que des versions plus récentes du JRE seront publiées, conformément aux recommandations ;
– pour activer (valeur 1) ou non (valeur 0) Java dans les navigateurs Web (c'est à dire le module complémentaire Java pour Internet Explorer et Mozilla Firefox) et Java Web Start, créer la ligne `WEB_JAVA=0/1` ;
– pour empêcher un redémarrage du poste de travail après l'installation, positionner `RebootYesNo` à No (ou à Yes le cas échéant) ;

Il est toutefois rappelé que ces paramètres deviennent inutiles lorsqu'une configuration centralisée par fichier `deployment.properties` (voir section 4.4) est mise en œuvre.

Les fichiers `.msi` et `.mst` doivent ensuite être déposés dans un partage de fichiers (d'un serveur de fichiers idéalement) accessible en lecture seule par tous les utilisateurs concernés. Le chemin UNC du partage créé sera utilisé dans la stratégie de groupe (c'est à dire par exemple `\\serveur\partage\jre1.8.0_92.msi` et non `C:\partage\jre1.8.0_92.msi` puisqu'il s'agit du chemin qui sera utilisé par les postes utilisateurs pour accéder aux fichiers d'installation).

Une simple stratégie de groupe permet pour finir de télédéployer le paquet d'installation. Au niveau de la console de gestion des stratégies de groupe du domaine, cela se fait en créant et modifiant une nouvelle GPO ayant les paramètres suivants :

Configuration ordinateur (activée) — masquer

Stratégies — masquer

Paramètres du logiciel — masquer

Applications attribuées — masquer

Java 8 Update 92 — masquer

Informations produit — masquer

Nom	Java 8 Update 92
Version	8.0
Langue	Anglais (États-Unis)
Plate-forme	x86
URL d'assistance	http://java.com/help

Informations de déploiement — masquer

Général	Paramètre
Type de déploiement	Attribué
Source du déploiement	\\[redacted]\Deployment\jre1.8.0_92_x586.msi
Désinstaller cette application lorsqu'elle se trouve en dehors de l'étendue de la gestion	Désactivé

Options de déploiement avancées	Paramètre
Ignorer la langue lors du déploiement de ce package	Désactivé
Rendre cette application 32 bits x86 disponible sur les ordinateurs de type Win64.	Activé
Inclure les classes OLE et les informations concernant le produit.	Activé

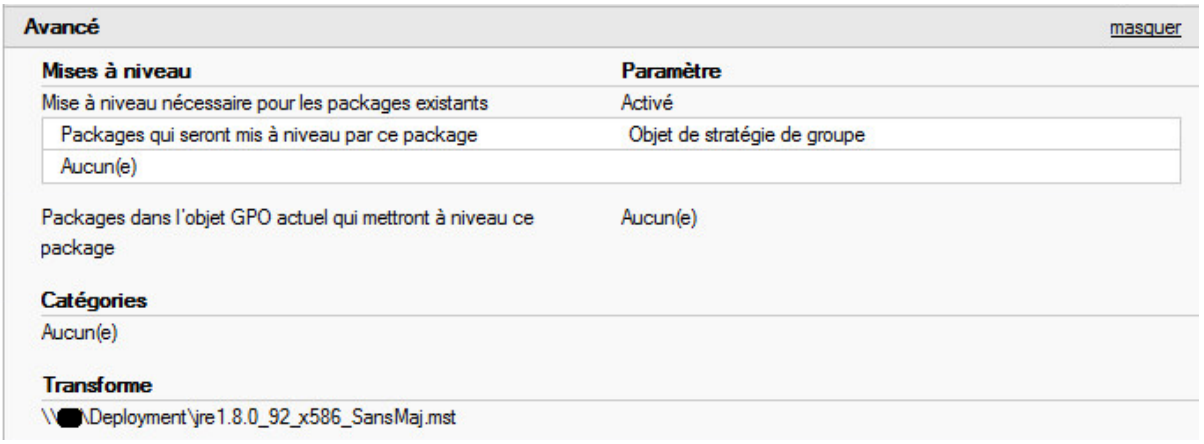

FIGURE 3 – GPO de déploiement de paquet d'installation MSI transformé

Le déploiement peut également se faire au niveau de la stratégie utilisateur plutôt que de la stratégie ordinateur. L'étendue d'application de la stratégie de groupe peut également être restreinte à certains groupes d'ordinateurs ou d'utilisateurs.

C.3 Déploiement par exécutable

Si le déploiement par exécutable est retenu, il est possible de spécifier les options d'installation par arguments de ligne de commande. Cette méthode est toutefois moins transparente qu'un déploiement par paquet MSI, cette dernière ayant également pour avantage de pouvoir gérer facilement les désinstallations ainsi que les mises à jour.

Les arguments pris en charge par l'exécutable d'installation du `JRE 8 Update 92` sont entre autres :

- `WEB_JAVA=0/1`, qui active (valeur 1) ou non (valeur 0) Java dans les navigateurs Web (c'est à dire le module complémentaire Java) et Java Web Start ;
- `WEB_JAVA_SECURITY_LEVEL=VH/H`, qui indique le niveau de sécurité du module complémentaire Java lorsqu'il est activé (`VH` pour « très élevé », `H` pour « élevé ») ;

Les différents niveaux de sécurité correspondent aux comportements suivants :

- au niveau « élevé », aucune appliquette non signée ne sera exécutée. Le certificat de signature doit qui plus est provenir d'une autorité autorisée, mais le statut de révocation du certificat peut ne pas être vérifié.
- au niveau « très élevé », aucune appliquette non signée ne sera exécutée. Le certificat de signature doit provenir d'une autorité autorisée, et le statut de révocation du certificat doit pouvoir être vérifiée préalablement à l'exécution.

Les modes `L` (« bas ») et `M` (« moyen ») ne sont plus disponibles.

Il est toutefois rappelé que la configuration centralisée par fichier `deployment.properties` (voir section 4.4) est la méthode à préférer. Le niveau de sécurité spécifié lors de l'installation par fichier exécutable n'est en effet pas verrouillé, il ne garantit donc pas contre une modification ultérieure, et ne représente qui plus est qu'une infime partie des paramètres de configuration du JRE.

Un script d'installation silencieuse pourrait alors ressembler à ceci :

```
@echo off
cls
echo .
echo .                    Installation du Java Runtime Environment 32 bits en cours.
echo .                                          Veuillez patienter.
echo .          (Cette fenêtre se fermera automatiquement après l'installation)
echo .
REM Installation du JRE 32-bit
"\\serveur\partage\jre-8u92-windows-i586.exe" /s WEB_JAVA=0 WEB_JAVA_SECURITY_LEVEL=H
exit /B %EXIT_CODE%
```

Liste des recommandations

R1 Anticiper l'obsolescence annoncée des appliquettes Java . . . 7
R2 Recenser les applications Java . . . 8
R3 Recenser les installations de Java . . . 8
R4 Recenser les modules complémentaires Java inutiles . . . 9
R5 Désinstaller les JRE inutiles . . . 9
R6 Définir des exceptions pour les postes utilisateurs ayant besoin du JRE . . . 9
R7 Filtrer au niveau des serveurs mandataires . . . 10
R8 Désactiver les modules complémentaires Java dans les navigateurs . . . 10
R9 Désactiver Java dans les navigateurs . . . 10
R10 Définir des exceptions pour les postes utilisateurs ayant besoin de Java dans un navigateur 10
R11 Remplacer les appliquettes Java . . . 11
R11⁻ Migrer les appliquettes Java . . . 12
R12 Définir une configuration sécurisée de Java verrouillée . . . 13
R12⁻ Définir une configuration sécurisée de Java non verrouillée . . . 13
R13 Restreindre le périmètre de déploiement du JRE . . . 14
R14 Faire de la veille . . . 14
R15 Maintenir les installations logicielles à jour . . . 15
R16 Mettre à jour le JRE 32 bits sur les systèmes 64 bits . . . 15
R17 Considérer les extensions JAR et JNLP au même titre que des exécutables . . . 16
R18 Maintenir et maîtriser deux navigateurs . . . 16
R19 Intégrer les MCO et MCS du JRE dans les projets . . . 17