ActiveX

&

la sécurité

NB : ce fascicule fait partie d'un travail de diplôme sur le sujet « Technologie ActiveX & Visual Basic 6 ».
Les autres fascicules peuvent être demandés à fcomte@caramail.com.

Critiques, remarques, questions ? fcomte@caramail.com

1 – Technologie ActiveX et sécurité

1.1 Introduction

Les contrôles ActiveX peuvent être invoqués depuis des pages web au travers d'un langage de script ou directement par une balise HTML <OBJECT>. Si le contrôle n'est pas installé localement, il est possible de spécifier une URL où l'obtenir. Une fois obtenu, le contrôle s'installe automatiquement si le navigateur le lui permet. Une fois installé, il peut alors être invoqué à nouveau sans avoir à le recharger.

Les contrôles peuvent être signés ou pas. Un contrôle signé offre un degré de vérification plus élevé que si il ne l'est pas, car il certifie sa provenance (son auteur) ainsi que le fait qu'il n'a pas été modifié. Cependant, la signature ne garantit pas la bienveillance, la fiabilité, ainsi que la compétence du signataire, mais seulement le fait que le contrôle provient bien du signataire. Etant donné que les contrôles ActiveX ne s'exécutent pas dans un « bac à sable » comme Java, il est essentiel d'avoir confiance en l'auteur du contrôle (ce dernier point étant vrai pour tout code qui s'exécute directement sans passer par un « bac à sable »).

Les problèmes de sécurité relatifs à la technologie ActiveX ne peuvent être ignorés. Les contrôles ActiveX sont partie intégrante de beaucoup de programmes, et ils sont requis pour des fonctions essentielles dans bon nombre d'environnements. Bien que les priorités sécuritaires changent d'une entreprise à l'autre, et d'un utilisateur à l'autre, il est important de comprendre les compromis entre fonctionnalité et sécurité afin de prendre des décisions adéquates.

1.2 Trop d'idées reçues…

Bien que les contrôles ActiveX présentent des défauts flagrants de sécurité, beaucoup d'idées reçues circulent encore trop à leur sujet. Il est donc intéressant d'évaluer objectivement ces risques.

Idée reçue #1 : *Tous les contrôles ActiveX sont dangereux.*

Les objectifs ainsi que la qualité des contrôles varie énormément, comme n'importe quel type de logiciels. Il n'est pas possible de déterminer – à première vue – si un contrôle présente plus de risques qu'un autre, car ce genre de problème n'est pas inhérent au contrôle.

Idée reçue #2 : *Les contrôles ActiveX sont très différents des exécutables « normaux » .exe (ou : Les contrôles ActiveX sont très comparables aux exécutables « normaux » .exe).*

Les contrôles ActiveX partagent bon nombre d'attributs communs avec les fichiers exécutables ordinaires. Ils sont exécutés directement sur la machine ; ils sont généralement « opaque » ; ils sont écrits dans des langages de haut-niveau ; ils varient considérablement en fonctionnalités, en qualité, ainsi qu'en caractéristiques sécuritaires. De plus, contrairement aux fichiers exécutables ordinaires, les contrôles ActiveX sont « mobiles », et peuvent souvent être invoqués à distance.

Idée reçue #3 : *Les plus grands dangers de sécurité des contrôles ActiveX proviennent de contrôles hostiles tels Internet Exploder (voir bibliographie).*

Il est vrai que des contrôles ActiveX - intentionnellement malicieux – peuvent potentiellement occasionner de gros dommages. Néanmoins, en pratique, il existe assez peu de contrôles de ce type, et la technologie Authenticode de Microsoft fournit une protection relativement sûre pour contrer ce genre de contrôles.

Idée reçue #4 : *Tous les contrôles signés sont sûrs.*

Rien ne pourrait être moins vrai… La technologie Authenticode (signature digitale de Microsoft) fournit un haut degré de vérification sur le contrôle, permettant d'assurer que ce dernier est produit par le signataire et qu'il n'a pas été modifié par quelqu'un d'autre. Une signature digitale ne garantit en aucun cas la compétence ni la bienveillance de son auteur. C'est à l'utilisateur final (ou à l'administrateur) de décider si tel contrôle mérite confiance.

Idée reçue #5 : *En évitant d'utiliser Internet Explorer et Outlook, on évite les problèmes liés aux contrôles ActiveX.*

Beaucoup d'applications tierces utilisent à un moment donné un ou plusieurs contrôles ActiveX pour des opérations ordinaires. De plus, elles peuvent également exécuter des scripts, au sein desquels peuvent se loger des ActiveX. Ainsi, ce n'est pas en évitant d'utiliser des produits Microsoft qu'on se protégera d'attaques basées sur contrôles ActiveX.

1.3 Risques et ennuis liés à la sécurité

ActiveX est une technologie puissante mais toutefois potentiellement dangereuse. La plupart des risques peuvent être évités si l'on est conscient des problèmes. Nous allons donc voir ici en détail comment la sécurité doit être traitée et à quel niveau d'utilisation des contrôles elle intervient. Cette sécurité peut être divisée en deux grandes branches : celle concernant l'importation et l'installation de contrôles, et celle concernant l'exécution de contrôles. Cette dernière branche est également divisée en deux sections : les ennuis liés directement aux contrôles, et ceux liés au scripting.

1.3.1 Ennuis liés au téléchargement – importation et installation de contrôles

1. Idéalement, la décision d'installer un logiciel doit être prise sur la base des fonctionnalités de ce dernier. Ce n'est pas le cas pour un contrôle ActiveX. Au lieu de cela, la décision est basée sur la source (présumée) du contrôle, ce qui est insatisfaisant pour deux raisons. Premièrement, le signataire du contrôle peut ne pas être plus capable d'assurer la sûreté du contrôle que l'utilisateur final lui-même. Deuxièmement, l'utilisateur final doit faire confiance dans la chaîne de distribution de la source originale du signataire.
   Le même problème se pose avec tous les systèmes de protection à signature : des contrôles sûrs peuvent provenir de sources non sûres, et des contrôles non sûrs peuvent provenir de sources sûres.

2. La signature d'un contrôle ActiveX est persistante : une fois que le contrôle a été signé par un organisme tiers digne de confiance, il est « ouvert » aux attaques. Si une vulnérabilité est découverte dans un contrôle signé, les « pirates » peuvent utiliser la relation de confiance entre la victime et le signataire pour introduire un contrôle vulnérable. Ainsi, un contrôle signé mais vulnérable fournit un bon moyen aux « pirates » d'installer quelque logiciel au sein de la machine de l'utilisateur, en arrivant à le convaincre qu'il a affaire à un organisme de confiance.

3. Dans l'architecture actuelle Windows, un contrôle ne nécessite de ne se faire enregistrer qu'une seule fois par machine. Cela peut poser problème si la machine est partagée par plusieurs utilisateurs : si l'un d'entres eux télécharge un contrôle, il est disponible par tous les autres.

4. Il n'est pas toujours possible de désinstaller un contrôle, ni d'implémenter des fonctionnalités sécuritaires.

1.3.2 Ennuis liés à l'exécution de contrôles

1. Les contrôles ActiveX ont plus de « pouvoirs » que les outils qui s'exécutent uniquement dans un bac à sable. Les contrôles ActiveX, puisqu'ils sont en code natifs s'exécutant directement sur la machine, sont capables d'accéder à des services et à des ressources non disponibles pour du code s'exécutant dans un environnement restreint.

2. Quasiment tous les mécanismes de sécurité liés aux contrôles ActiveX sont basés sur Internet Explorer. Malheureusement, les contrôles, eux, ne sont pas liés uniquement à Internet Explorer, ils peuvent être installés et exécutés indépendamment de ce logiciel. Des organismes tiers peuvent donc ne pas garantir les mécanismes de sécurité disponibles dans Internet Explorer.

3. Bien des mécanismes fournis par Internet Explorer sont primitifs. Quelques-uns sont des propositions du style « tout ou rien », forçant l'utilisateur à choisir entre fonctionnalité et sécurité. Par exemple, il n'existe pas de moyen d'exécuter un contrôle « non sûr pour le scripting » sans avoir à autoriser tous les contrôles « non sûrs pour le scripting ».

4. Lorsqu'un contrôle est exécuté, il utilise généralement les privilèges de l'utilisateur en cours. Il n'existe pas de mécanisme de restrictions de privilèges (tel celui d'un bac à sable).

5. Parce que les contrôles ActiveX peuvent être invoqués de manière distante (au travers d'une page HTML ou d'un email), chaque contrôle offre un canal dans un réseau qui peut potentiellement être utilisé par un pirate.

6. Les contrôles ActiveX sont difficiles à gérer et à auditer, particulièrement pour des utilisateurs non experts. Les outils permettant de la faire manquent de fonctionnalités.

1.3.3 Ennuis liés au scripting

1. Les contrôles ActiveX liés au scripting (VBScript ou Javascript) sont responsables de l'implémentation de leur sécurité. Puisque les contrôles ne s'exécutent pas dans un environnement restreint (bac à sable), chaque contrôle utilisant du script implémenter son propre bac à sable, ce qui est extrêmement compliqué. De plus, ces contrôles doivent assurer un bon fonctionnement quel que soit les entrées fournies, dans quel ordre que ce soit.

2. Un script peut faire appel à un contrôle d'une manière que le développeur de celui-ci n'a pas prévu. Cela peut mener à un comportement incorrect qui pourrait être exploité pour violer les règles de sécurité.

3. Un script peut utiliser un contrôle de manière translucide. L'utilisateur peut ne pas s'en rendre compte, et il pourrait alors être impossible de déterminer au préalable quel contrôle est utilisé (dans une page HTML). Dans un tel cas, l'utilisateur ne peut tout simplement pas prendre de décision concernant l'ouverture ou non d'un tel document.

4. ActiveX est un moyen pour les scripts d'une page web d'échapper au bac à sable d'Internet Explorer. On pourrait dès lors imaginer l'utilisation d'un contrôle ActiveX pour contourner les règles de sécurité.

5. Les moteurs de script autres que celui d'Internet Explorer peuvent ne pas fournir de mécanisme de sécurité tel que le fait IE.

1.4 Bénéfices d'ActiveX et fonctionnalités sécuritaires

L'utilisation des contrôles ActiveX présentent autant de bénéfices que de risques. Les fonctionnalités sécuritaires du système Windows peuvent aider à gérer les contrôles. Nous allons voir maintenant quelles sont les possibilités de tels mécanismes.

1.4.1 Bénéfices d'ActiveX

Citons parmi les principaux bénéfices :

Les contrôles ActiveX promeuvent la réutilisation.
Puisque l'invocation d'interfaces entre les contrôles est standardisée, les développeurs de systèmes et d'applications peuvent facilement réutiliser les contrôles pour d'autres buts, indépendamment du choix du langage ou de l'environnement de développement.

Les contrôles ActiveX offrent plus de fonctionnalités que d'autres outils qui s'exécutent uniquement dans un bac à sable.
Puisque les contrôles ActiveX sont formés de code natif s'exécutant directement sur la machine, ils sont capables d'accéder aux services et ressources qui ne peuvent l'être par des applications s'exécutant dans un environnement restreint. Ceci permet de réaliser des choses qui ne pourraient être faites dans un tel environnement.

Les contrôles ActiveX fournissent un large spectre de possibilités.
Pour bien des entreprises, les contrôles ActiveX représentent le moyen le plus simple, le plus puissant et le plus économique de répondre à des problèmes spécifiques.

1.4.2 Fonctionnalités sécuritaires d'ActiveX

Il existe une quantité d'outils permettant de gérer convenablement la sécurité avec ActiveX. Citons :

Les réglages « Approuvé par l'administrateur » - de la zone de sécurité d'Internet Explorer. Il s'agit d'une option permettant d'exécuter uniquement les contrôles qui ont été approuvés par la personne qui administre le système.

La technologie Authenticode – utilisée pour signer digitalement les contrôles et vérifier le contenu exécutable, mais aussi de contrôler le téléchargement du code entre le serveur et le client.

Le CodeBaseSearchPath – Avec cette clé de registre, les administrateurs peuvent contrôler où le système est accédé lorsque le contrôle est téléchargé et installé.

L'outil Internet Explorer Administration Kit (IEAK) – Il s'agit d'un outil qui peut être utilisé par les administrateurs réseau ou par les ISP (Internet Services Providers) pour ajuster et contrôler les réglages d'IE de manière centralisée, les distribuer aux utilisateurs, et administrer les réglages d'Authenticode.

L'interface IObjectSafety – fourni une méthode qui, en plus des fanions (flags) « Safe For », détermine en quoi un contrôle ActiveX est sûr.

Les fanions « Safe For » - les fanions « Safe For Initialization » et « Safe For Scripting », généralement fixés lors de l'installation d'un contrôle ActiveX, déterminent les actions qu'une page web peut se permettre lorsqu'elle rencontre un contrôle ActiveX.

Le « Kill Bit » - il s'agit d'une valeur du registre qui interdit à IE de charger un contrôle. Elle ne peut être écrasée par aucune autre valeur dispensée par une configuration différente des zones de sécurité d'IE.

Les zones de sécurité – d'IE (et d'autres produits dans lesquels la sécurité est importante) peuvent être utilisées pour vérifier fermement le comportement et l'accessibilité d'un contrôle. IE inclus la possibilité de grouper les sites web dans quatre groupes différents, plus une zone nommée « mon ordinateur », qui doit être configurée à l'aide de IEAK.

Windows 2000 Group Policy Objects (GPO) – tout comme IEAK, le GPO est utilisé pour gérer les configurations d'IE ainsi que d'Authenticode de manière centralisée.

1.5 Authenticode

Microsoft Authenticode est le nom d'un ensemble de technologies pour la signature digitale et la vérification de contenu exécutable. Une personne qui développe un contrôle ActiveX ou une applet Java peut signer de façon digitale le fichier cabinet (.CAB) en utilisant la clé privée du certificat digital. Ce certificat peut être obtenu depuis un CA (Certificate Authority) commercial, comme Verisign, GTE ou Thawte ou depuis un CA d'entreprise comme les services de certification de Windows 2000.

L'outil de signature de code de Microsoft signcode.exe est utilisé pour signer le programme ; il requiert du signataire un certificat stocké dans l'API Microsoft Crypto. Les certificats et leurs clés privées associées sont automatiquement stockées dans cette API si elle sont sollicitées depuis une page web en utilisant IE. Si le certificat est obtenu par un autre moyen, il sera chargé dans l'API en utilisant le format PKCS#12 (fichier d'extension .p12 ou .pfx).

L'outil signcode.exe ne signe des fichiers que si le certificat utilisé contient une valeur donnée dans la clé « Enhanced Key Usage ». Cet outil est disponible à l'adresse http://msdn.microsoft.com/downloads, sous la rubrique Security/Authenticode.

Internet Explorer valide automatiquement la signature digitale du code mobile lorsque l'utilisateur télécharge le fichier .CAB associé. Une signature valide donne à l'utilisateur le nom de la personne qui a créé le code et l'assure que le code n'a pas été altéré depuis cette signature. Ceci permet à l'utilisateur de se poser la question « Puis-je faire confiance à la société/personne qui a créé ce code de telle manière qu'il ne transporte ni virus ni bugs ? ». Si l'on considère la méthode traditionnelle d'achat de logiciels comme se déplacer dans un magasin et acheter une boîte en carton, la signature digitale peut être comparée comme une boîte sur laquelle figure l'autocollant « Certificate of Authenticity »… Le fait que le programme soit signé ne veut pas dire qu'il soit de meilleure qualité que celui qui est dans une boîte sans autocollant !

Les utilisateurs ne veulent pas tous avoir à prendre cette décision à chaque fois qu'ils installent un nouveau programme, et dans un environnement d'entreprise il est judicieux de mettre en place une autorité centrale qui établit les règles d'installation de logiciels. IEAK permet aux administrateurs de configurer ces règles et de développer des polices de sécurité relatives à IE qui peuvent être déployées ensuite sur les postes des utilisateurs. Ce genre de configuration interdit à ceux-ci de télécharger et d'exécuter du code peu sûr, mais autorise le code signé par des auteurs reconnus. Par exemple, il serait possible d'établir un processus qui appelle tout code développé de manière interne afin que celui-ci soit vérifié avant déploiement. Ce processus pourrait même inclure une révision du code permettant d'assurer que le programme ne soit pas utilisé de manière malicieuse. Une fois ce code approuvé pour le déploiement, l'outil de configuration associé au processus pourrait le signer. Pour renforcer cette politique, tous les navigateurs de l'entreprise devraient être configurés de manière à n'exécuter que le code signé par l'outil de configuration interne à la compagnie.

Les applets Java s'exécutant dans une machine virtuelle au sein du navigateur, IE - couplé avec la technologie Authenticode - peut être utilisé pour définir le niveau de confiance requis pour exécuter les applets. L'auteur signataire de l'applet peut fixer dans la signature quelles sont les permissions que celle-ci nécessite[1].

[1] La liste des permissions disponibles avec laquelle une applet Java peut être signée est disponible à l'adresse http://www.microsoft.com/java/sdk/40/pg/pg_pkgdist_sgncd.htm.

Le processus de certification disponible chez Verisign coûte US$ 400 par année pour les logiciels commerciaux (classe 3) et US$ 20 par année pour les logiciels créés par des particuliers.

1.6 CodeBaseSearchPath

La plupart des pages web qui utilisent des contrôles ActiveX fournissent au client des informations quant à la localisation du code à télécharger et à installer. Cette information est donnée dans l'attribut `CODEBASE` d'une balise `OBJECT`, et référence le composant lui-même ou le fichier CAB contenant le contrôle et les instructions d'installation (fichier .inf).

Par exemple, le code HTML suivant provient de la homepage du site investor.msn.com :

```
<OBJECT type= « application/x-oleobject »>
<classid= "clsid :62360003-D8A7-418b-9DC6-2B9DE95273A0"
codebase=http://fdl.msn.com/public/investor/v8/0326/ticker.cab#version=8,20
00,0326,2 width=100% height=34>
</OBJECT>
```

La balise `OBJECT` est utilisée pour charger un objet dans une page HTML (ici un contrôle ActiveX). Lorsqu'IE lit le code HTML, il va d'abord vérifier si l'objet est déjà installé sur la machine, grâce au `classid` spécifié. Si le numéro ne peut être trouvé dans la base de registres, par défaut IE regarde sous l'attribut `codebase`. Dans notre exemple concret, le contrôle est encapsulé dans `ticker.cab` et sera téléchargé depuis le site `fdl.msn.com`.

2 – Technologie Java et sécurité

L'exécution de code inconnu sur un client étant une opération qui peut s'avérer extrêmement dangereuse, la sécurité constitue un des aspects les plus essentiels de Java. Le langage protège l'utilisateur en ce sens qu'il s'interdit en principe tout accès aux ressources locales, ce qui devrait théoriquement mettre l'utilisateur à l'abri de malveillance ; de plus, le confinement de Java dans une machine virtuelle est lui aussi garant de sécurité[2].
Il est néanmoins vrai que pour diffuser des applets non triviales sur le web, il faut être en mesure d'accéder aux ressources locales ; ainsi, une application de telebanking doit pouvoir accéder au fichier DTA (une norme pour le transfert de fichiers de paiement entre un client et sa banque) que l'utilisateur a préparé offline en vue du paiement de ses factures mensuelles. Ceci nécessite bien sûr une relaxation des règles très strictes mises sur pied par Java ; d'un autre point de vue, une telle relaxation est extrêmement dangereuse. Il faut donc définir une stratégie permettant cette relaxation de manière raisonnablement sûre.

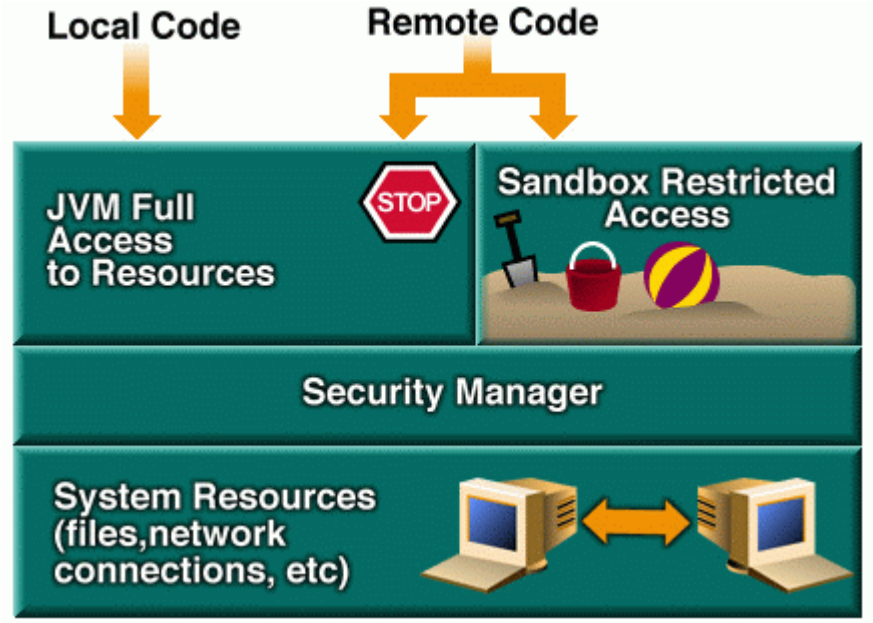
Il s'agit d'une opération relativement complexe dont nous nous contenterons de détailler quelques-unes des possibilités offertes par Java.

2.1 L'évolution de la sécurité

La sécurité – de manière générale – est renforcée dans Java à travers une série de mécanismes : premièrement, le langage a été pensé pour être relativement simple et « type-safe ». Ainsi, des fonctionnalités telles la gestion automatique de la mémoire, un « garbage collector « (ramasse-miettes), une vérification de taille sur les tableaux et les chaînes sont des exemples permettant d'aider notablement le développeur. Deuxièmement, les compilateurs ainsi qu'un vérificateur de bytecode assure que seul du bytecode légitime est exécuté. Le vérificateur, avec la machine virtuelle Java, garantit la sécurité lors de l'exécution. De plus, un loader de classe (*Classloader*) définit un espace local de nommage, utilisé pour assurer qu'une applet non-sûre (*untrusted*) ne puisse interférer avec l'exécution d'autres programmes.
Finalement, l'accès aux ressources critiques du système est régularisé par la JVM et vérifié à l'avance par la classe Security Manager, qui limite les actions d'un code peu sûr au strict minimum.

Le modèle de sécurité proposé par Java a évolué au cours des diverses versions de la spécification. Le modèle original, appelé aussi « bac à sable[3] » (*sandbox model*) donnait un environnement très contraignant dans lequel un code non certifié comme sûr ne pouvait s'exécuter.

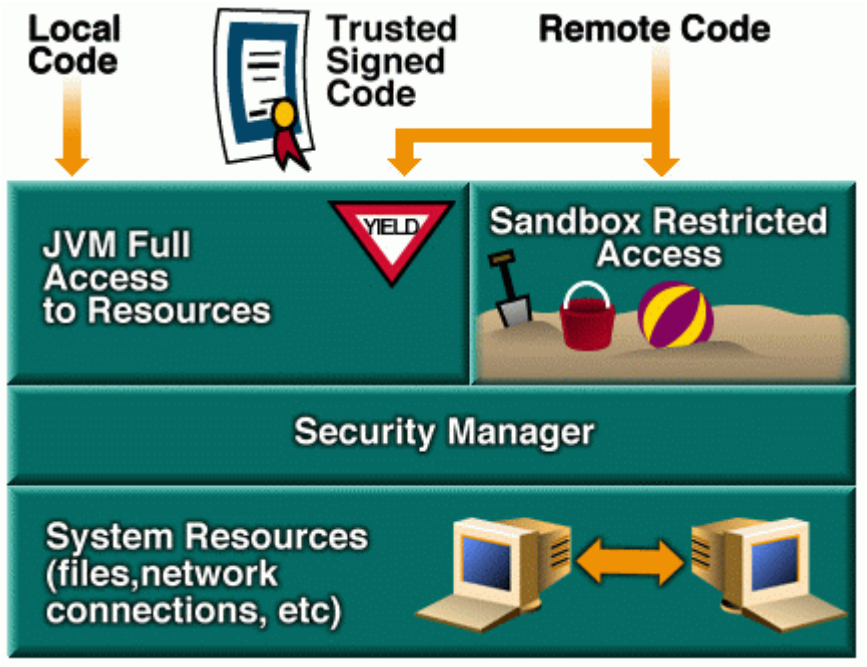
[2] Ce confinement est plus important qu'il n'y paraît : Microsoft a, dans une version intermédiaire de sa propre machine virtuelle Java, autorisé l'accès direct aux ressources écrans (par DirectX) à la machine virtuelle Java. Peu de temps après, un producteur de logiciel Java indépendant (Anfy Java) parvenait (malencontreusement) à crasher toute machine Windows 95 ou 98 avec une applet relativement triviale…
[3] Le nom de bac à sable provient du fait que le modèle permet en effet de faire de nombreuses choses avec Java localement, mais rien n'est permanent : les constructions des applets téléchargées s'apparentent de ce fait à des châteaux de sables…



Modèle de sécurité de la JVM (Java Virtual Machine) 1.0

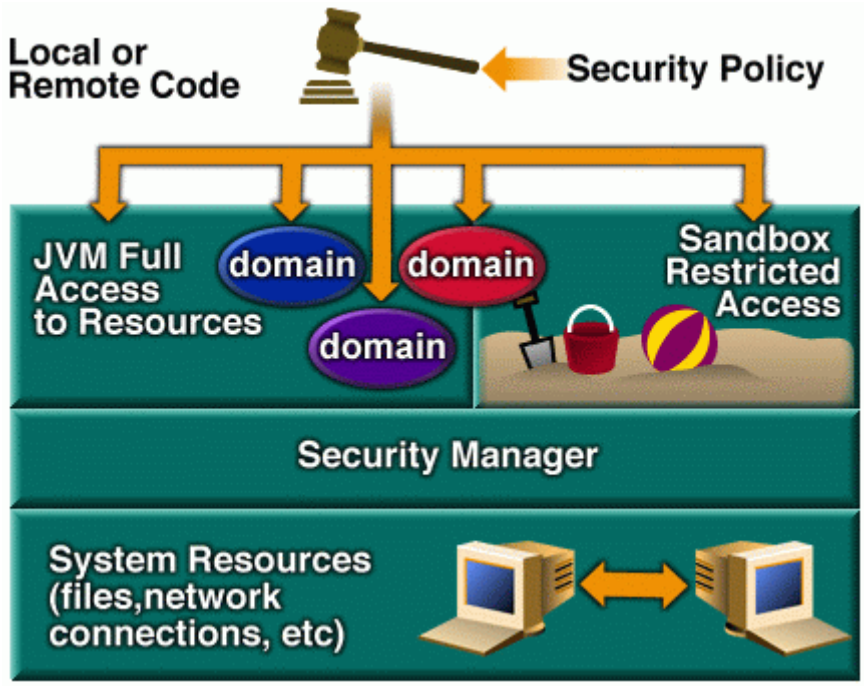
Ce modèle sous-entendait que du code téléchargé d'un serveur distant était potentiellement peu sûr (*untrusted*) alors que le code local était sûr (*trusted*).

Le JDK 1.1 introduisait le concept d'applets « identifiées » (*signed applets*). Une applet identifiée peut être traitée comme du code local, dans la mesure où la clé publique utilisée pour vérifier la signature de l'applet est digne de confiance. Des applets identifiées sont en principe fournies sous la forme de fichiers archive Java (*Java ARchives files*).



Modèle de sécurité de la JVM 1.1

La principale nouveauté qu'introduisit JDK[4] 1.2 est l'existence de règles de sécurité (*security policy*). Que le code exécuté soit de provenance locale ou non, il est soumis à des règles de sécurité qui définissent individuellement, pour chaque provenance, le sous-ensemble d'opérations que le code pourra effectuer sur la machine locale. Ce sera en principe le rôle de l'administrateur système de définir quelle provenances disposeront de quels privilèges. Cette définition peut être effectuée de manière relativement flexible, en séparant les sources en domaines, par exemple.



Modèle de sécurité de la JVM 2.0

[4] Java Development Kit, permet de développer des applications et des applets Java.

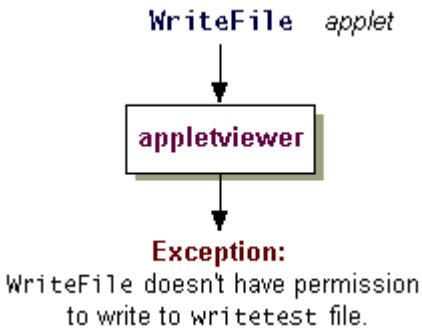
2.2 La sécurité dans le JDK 1.2

JDK 1.2 introduit trois nouveaux outils :

- `Keytool`, utilisé pour créer des paires de clés publiques et privées, importer, exporter et afficher des certificats, et générer des certificats de la norme X.509,
- `Jarsigner`, un outil qui permet de signer des fichiers JAR et de vérifier l'authenticité de la signature,
- Et les règles de sécurité (`Policy Tool`).

Le Security Manager de Java constitue la méthode normale de protection du client contre des attaques de virus par des applets hostiles. En l'absence de toute intervention de la part de l'utilisateur du client, une applet n'a pas accès aux ressources locales.

Ainsi, une applet qui désire tout simplement ouvrir un fichier en écriture et y écrire ne peut le faire[5], car elle n'a, par défaut, pas le droit d'accéder aux ressources locales. L'exécution d'une telle applet (à la ligne de commande par exemple, avec appletviewer) renvoie une exception, comme celle figurant sur la figure suivante :

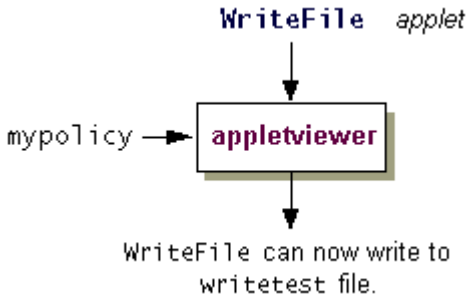


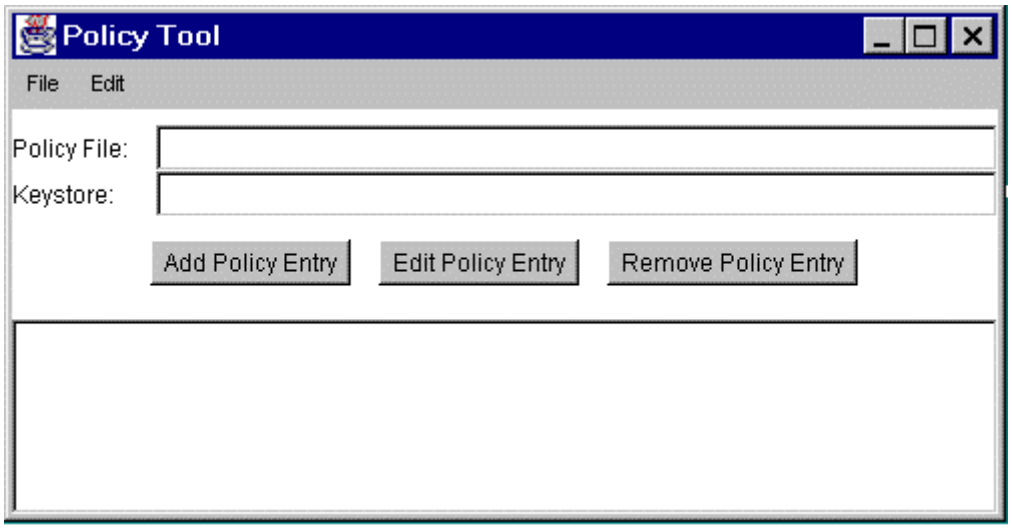
Il est possible de mettre sur pied de nouvelles règles de sécurité auxquelles seront soumis tous les programmes arrivant sur la machine de l'utilisateur. Le fichier de règles est un fichier ASCII pouvant être établi à l'aide de tout éditeur de texte, ou plus confortablement au moyen d'un outil de configuration permettant une programmation plus aisée, et vérifiant que la syntaxe utilisée dans le fichier soit correcte.

La marche à suivre pour cette manœuvre est très simple. L'utilitaire se lance à la ligne de commandes avec `policytool`, ce qui ouvre la fenêtre de l'utilitaire de configuration de règles. Nous ne détaillerons pas ici tout le processus, mais notons qu'il est possible de définir un ensemble de règles (accès simple, lecture, écriture) sur une URL donnée (en fait un codebase, soit l'url du code téléchargeable[6]). Cet ensemble peut ensuite être sauvegardé dans un fichier (`mypolicy`, par exemple), qui sera lu lors d'un accès à du code téléchargeable.

[5] Le fichier exemple est disponible sur le site de Sun à l'adresse http://java.sun.com/docs/books/tutorial/security1.2/tour1/example-1dot2/WriteFile.java.
[6] Une URL du type http://java.sun.com/docs/books/tutorial/security1.2/- permet par exemple d'accorder la permission à tout code en provenance de l'ensemble du répertoire security1.2.

Si l'on donne le droit de créer un fichier et d'y écrire par exemple, l'exception levée auparavant n'apparaîtra plus, comme le montre la figure suivante.





L'outil de configuration des règles de sécurité de Java

Notons que Java fournit également des extensions pour la cryptographie (Java Cryptographic Extensions JCE), contenant plusieurs implémentations d'algorithmes de signature électronique (DSA), un algorithme de signature de message (SHA-1), des interfaces de certifications, etc.

3 - Synthèse des questions sécuritaires

ActiveX

Les contrôles ActiveX sont supposés être sécurisés en étant signés digitalement avec un code d'authentification identifiant son propriété. Cela permet à l'utilisateur de choisir de télécharger, par exemple, un contrôle écrit par une source « sûre » (Microsoft par exemple), mais pas un contrôle écrit par une société inconnue…
L'auteur de l'ActiveX doit obtenir un certificat pour être capable de signer un contrôle. Mis à part le problème du choix des auteurs dignes de confiance, il y a le danger qu'un auteur de pages Web sans scrupules exploite les faiblesses d'un contrôle légitime.

Java

Le gestionnaire de sécurité intégré à Java est consulté chaque fois qu'une application Java essaie d'accéder le disque, les ports réseaux et les processus externes. Cet outil constitue la méthode normale de protection du client contre des attaques de virus (ou autres…) par des applets hostiles. Dans un navigateur Internet, le gestionnaire interdit la plupart des types d'accès si bien qu'une applet Java téléchargée depuis Internet ne peut pas causer de dommages (en l'absence de toute intervention de la part du client). En contre partie, cette sécurité peut également empêcher l'applet d'accomplir des tâches utiles. A partir de la version 1.1 de Java, les applets peuvent être signées de la même façon que les contrôles ActiveX.

Quelques mots sur les plug-ins

Les plug-ins sont des programmes qui ajoutent du code au navigateur Internet. Le modèle de sécurité d'un plug-in est le même que pour ActiveX : lorsqu'on en télécharge un, on place sa confiance dans le constructeur du plug-in. De même, tous les avertissements relatifs à ActiveX sont valables pour les plug-ins : ils peuvent être très dangereux du point de vue de la sécurité. En acceptant un plug-in comme Shockwave (de Macromedia), il n'est donc pas certain que des problèmes sécuritaires n'apparaissent pas ; à nouveau, ce n'est pas parce qu'un programme est censé être exempt de bugs qu'il n'en a pas…
Certains spécialistes (comme ceux de l'équipe « Secure Internet Programming », de l'Université de Princeton) estiment qu'il « suffit » d'autoriser le téléchargement de plug-ins de fonctionnalités générales (audio, ou vidéo par exemple), permettant une exploitation (potentiellement néfaste) moindre des ressources sensibles sur la machine cliente. Sans être trop paranoïaque, d'autres personnes (l'auteur de ce rapport en fait partie) utiliseront des solutions logicielles de Firewall et autres mécanismes de protections, permettant de détailler les informations transitant entre la machine client et le réseau.

Bibliographie et liens critique

- JAVA deuxième partie : les extensions en télécommunications, cours de Markus Jaton, professeur de téléinformatique à l'EIVD, version du 27 janvier 2001.
  Le cours de Mr. Jaton m'a été utile pour confronter le modèle de Microsoft à celui d'un de ses concurrents, RMI.

- http://www.microsoft.com/com/tech/activex.asp, la page officielle de Microsoft pour la technologie ActiveX. On y trouve les « White papers », une sélection d'ouvrages sur le sujet, ainsi que divers articles de presse.

- http://www.cs.princeton.edu/sip/java-vs-activex.html, un article de l'équipe « Princeton Secure Internet Programming Team » comparant les technologies ActiveX et JAVA du point de vue de la sécurité.

- http://www.cert.org/reports/activeX_report.pdf
  Les 22 et 23 août 2000, un centre de coordination du CERT (un centre d'expertise de la sécurité sur Internet), composé d'une vingtaine de personnes issues d'un large éventail d'entreprises (AT&T, DoD, Microsoft, NSA, …) s'est réuni afin de discuter des problèmes de la sécurité relativement aux contrôles ActiveX. Le résultat de leur travail est disponible sous la forme d'un fichier .pdf de 49 pages, couvrant aussi bien les risques liés à la sécurité que les suggestions faites pour contrer ceux-ci.

- http://www.users.zetnet.co.uk/hopwood/papers/compsec97.html
  Un travail de comparaison de David Hopwood sur la sécurité entre Java et ActiveX, présenté lors de la 14ème conférence sur la sécurité informatique en 1997.

- http://www.halcyon.com/mclain/ActiveX/welcome.html
  Le site personnel d'un développeur américain qui a démontré en 1996 que la technologie ActiveX souffrait de trous de sécurité énormes : son contrôle, une fois activé, éteins la machine proprement…

- http://java.sun.com/docs/books/tutorial/security1.2/overview/index.html
  Un résumé de l'évolution de la sécurité sous Java.

Autres articles relatant la sécurité et ActiveX

- http://www.webdeveloper.com/activex/activex_security.html

- http://www.webdeveloper.com/security/security_java_activex.html

- http://news.cnet.com/news/0,10000,0-1005-200-316425,00.html

- http://www.cs.uwindsor.ca/units/library/talks/security/index.html

- http://news.cnet.com/news/0,10000,0-1003-200-317102,00.html

- http://ibelgique.ifrance.com/secur/p48.html