Examen d'*Initiation à la sécurité*

Université Paris 12, Master d'Informatique, 1ère année

1ère session, 21 mai 2007

Transparents de cours autorisés
Pas de notes manuscrites, pas de corrections de TD/TP

Exercice 1: [2pts] Modéliser en tant que système de contrôle d'accès la situation suivante :

- Le système est composé de trois types d'objets : fichiers, agents, et superviseurs.
- Les agents ont le droit de lire seulement dans les fichiers qu'ils possèdent.
- Un agent seul ne peut écrire dans un fichier que s'il a le droit d'écriture dans ce fichier et s'il est accompagné par un autre agent (on parle alors d'*agents mutuellement authentifiés*).
- Un agent X peut céder à un 2e agent Y son authentification mutuelle auprès d'un autre agent Z, si le superviseur considère Z de confiance.
- Un agent X de confiance (pour le superviseur) peut créer un nouveau agent qui sera aussi de confiance.

Exercice 2: [3pts]

1. Écrire un `SecurityManager` qui, une fois installé, permettrait à toute classe de lire seulement les fichiers dans le répertoire `/home/etudiant/examen` qui ont l'extension `java`, (pas de droit de lecture pour les autres fichiers dans ce répertoire).
2. Résoudre le même problème avec une `policyfile`.

Exercice 3: [9pts] Écrire deux programmes Java qui devraient se comporter de la manière suivante :

1. Le premier (Alice) devrait générer une clé de session pour l'algorithme DES et la transmettre au 2e (Bob). La clé devrait être cryptée par la clé publique de Bob et accompagnée par la signature par la clé privée de Alice.
2. Bob devrait récupérer l'envoi d'Alice, décrypter la clé de session, vérifier la signature, et ensuite envoyer un message à Alice, crypté avec la clé de session.
3. Enfin, Alice devrait récupérer le message de Bob, le decrypter et l'afficher à la sortie standard.

On suppose que chacun des agents garde sa propre clé priviée ainsi que la clé publique de l'autre (toutes de type RSA) dans un Keystore.

Enfin, vous pouvez utiliser un paramètre de type `InputStream` et un autre de type `OutputStream`, externes à chacune des deux classes, pour les communications entre les agents. (C'est à dire, on ne s'intéresse pas et on ne note pas l'implémentation de la transmission des messages, que ce soit par sockets ou par fichiers ! On ne s'intéresse qu'au bon enchaînement des opérations de chiffrement/déchiffrement.)

Exercice 4: [6pts] Considérons le protocôle suivant, dans lequel on considère que A et B possèdent K_S :

$$\begin{aligned}
A \longrightarrow S &: A, B \\
S \longrightarrow B &: \{\{K_1\}_{K_S^{-1}}\}_{K_A}, A, B \\
B \longrightarrow A &: \{\{K_1\}_{K_S^{-1}}\}_{K_A}, A, B \\
A \longrightarrow B &: \{M, N_A\}_{K_1}, A, B \\
B \longrightarrow A &: N_A, A, B
\end{aligned}$$

1. Pourquoi ce protocôle n'est pas correctement construit ? Donnez une modification de la 2e ligne qui fait que le protocôle soit correctement construit.
2. Enoncez une assertion d'authenticité qui n'est pas satisfaite par ce protocôle, et donnez une attaque qui prouve l'incorectitude de l'assertion.
3. Donnez une correction du protocôle qui satisfera l'assertion énoncée.
