Sécurité Informatique

Manuel Munier

UPPA STEE - IUT des Pays de l'Adour - Département RT - **LIUPPA**
manuel.munier@univ-pau.fr
https://munier.perso.univ-pau.fr/teaching/m2ti-ssi/

Master Technologie de l'Internet
2023-2024

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR
COLLÈGE STEE
SCIENCES ET TECHNOLOGIES POUR L'ÉNERGIE ET L'ENVIRONNEMENT

iut réseaux & télécoms Mont de Marsan

Préambule

- Il faut améliorer la qualité des systèmes informatiques
  - les pannes informatiques paraissent insupportables
  - → les utilisateurs se sont habitués à une "informatique invisible" permanente
  - → les experts parlent d'un monde technologique que l'on ne peut plus débrancher...
- Importance économique du logiciel
  - importance croissante de l'informatique dans l'économie
  - coût du logiciel supérieur à celui du matériel
  - coût de la maintenance supérieur à celui de la conception

Pb : Démarche ingénierie encore mal intégrée

⇒ la non qualité des systèmes informatiques a des conséquences qui peuvent être très graves

Pannes logicielles

- Convocation de centenaires à l'école
  - convocation à l'école de personnes âgées de 106 ans
  - cause : codage de l'année de naissance sur 2 caractères
- Mission Vénus
  - passage à 5 000 000 de km de la planète, au lieu de 5000 km prévus
  - cause : remplacement d'une virgule par un point (format US des nombres)
- 2 jours sans courant pour la station Mir (14→16 nov. 1997)
  - cause : plantage d'un ordinateur l'orientation des panneaux solaires

Pannes logicielles

- Passage de la ligne
  - au passage de l'équateur un F16 se retrouve sur le dos
  - cause : changement de signe de la latitude mal pris en compte
- Y2K : le bug de l'an 2000
  - la lutte contre le bug de l'an 2000 a coûté à la France 500 milliards de francs
  - cause : la donnée "année" était codée sur 2 caractères, pour gagner un peu de place
- La carte bleue
  - le secret des cartes bancaires reposait essentiellement sur un algorithme qui a été publié sur un newsgroup !

Pannes logicielles

- Socrate
  - système de réservation de places Socrate de la SNCF
  - ses plantages fréquents, sa mauvaise ergonomie, le manque de formation préalable du personnel, ont amené un report important et durable de la clientèle vers d'autres moyens de transport
  - cause : rachat par la SNCF d'un système de réservation de places d'une compagnie aérienne, sans réadaptation totale au cahier des charges du transport ferroviaire

Pannes logicielles

- Échec du premier lancement d'Ariane V
  - au 1er lancement de la fusée Ariane V, celle-ci a explosé en vol
  - cause : logiciel de plate forme inertielle repris tel quel d'Ariane IV sans nouvelle validation ; Ariane V ayant des moteurs plus puissants, elle s'incline donc plus rapidement que Ariane IV pour récupérer l'accélération due à la rotation de la Terre
  - les capteurs ont bien détecté cette inclinaison d'Ariane V, mais le logiciel l'a jugée non conforme au plan de tir (d'Ariane IV), et a provoqué l'ordre d'auto destruction
  - en fait, tout se passait bien...
  - coût du programme d'étude d'Ariane V : 38 milliards de Francs

Pannes logicielles

- Perte de Mars Climate Orbiter, le 23 septembre 1999, après 9 mois de voyage
  - coût : 120 M$
  - cause : confusion entre pieds et mètres

- Microsoft raciste ?
  - le correcteur d'orthographe de Word proposait "anti-arabe" pour corriger "anti-stress"...

Pannes logicielles

- eBay indisponible
  - l'indisponibilité durant 22 heures du serveur web de eBay, site de vente aux enchères, a fait échouer plus de 2,3 millions d'enchères
  - dans un secteur où la compétitivité dépend plus que jamais du zéro-défaut face au consommateur, eBay a annoncé qu'elle remboursait les frais d'enregistrement des enchères en cours le jeudi 10 juin 1999, soit entre 3 et 5 millions de dollars
  - à Wall Street, le titre a chuté de plus de 9% (S&T Presse, 14 juin 1999)

Pannes logicielles

- Le micro-ordinateur menace-t-il la productivité ?
  - des milliers d'heures de travail sont perdues à essayer de faire faire à l'ordinateur ce qu'il devrait faire, ou de comprendre des messages d'erreur incompréhensibles...

Pannes logicielles

Logiciel

Définition d'un logiciel [a]

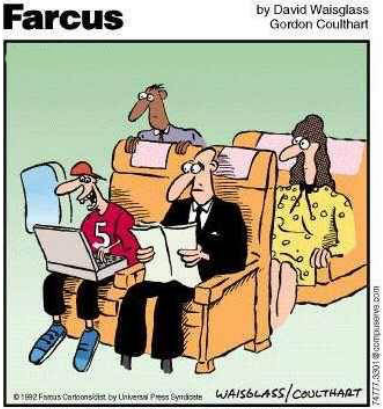
a. Arrêté ministériel du 22 décembre 1981

Ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de données (en anglais : *software*)

- Citation de Rich Cook :
  - → *"Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the universe trying to produce bigger and better idiots. So far, the universe is winning."*

"Qualité" des logiciels

- Les erreurs involontaires de conception et de codage représentent un tiers du coût des sinistres informatiques !
  - ⇒ Génie Logiciel
- La malveillance quant à elle cause 60% de ce coût...
  - ⇒ sécurité informatique



Yaahooo !!! J'ai réussi à faire planter le système de la tour de contrôle !!!

Sécurité informatique ?

- Différentes facettes de la sécurité informatique

⇒ Différentes compétences

⇒ Différents intervenants

MM Manuel Munier (MC LIUPPA - MdM)
⇝ *intro, politiques, sécurité Java, droit & numérique*

FK Frédéric Kustyan (CETE SO/DIM/CS - Bdx) [1]
⇝ *gestion des risques, audit sécurité*

1. Ministère de l'Écologie (CP2I/DOIP), Responsable Centre Serveur

Plan du cours

Plan du cours

La sécurité en informatique

- Intuitivement : permettre uniquement les actions légitimes, c'est-à-dire empêcher qu'un utilisateur puisse exécuter des opérations qui ne devraient pas lui être permises

⇒ Pour définir quelles sont les opérations autorisées et celles qui sont interdites, il faut établir une **politique de sécurité**

- Les ITSEC[2] (standard européen) définissent une politique de sécurité comme étant

> *"l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique"*

2. Information Technology Security Evaluation Criteria

La sécurité en informatique

- Pour construire une politique de sécurité il faut :
  - d'une part, définir un ensemble de propriétés de sécurité qui doivent être satisfaites par le système
    - → ex : *"une information classifiée ne doit pas être transmise à un utilisateur non habilité à la connaître"*
  - d'autre part, établir un schéma d'autorisation, qui présente les règles permettant de modifier l'état de protection du système
    - → ex : *"le propriétaire d'une information peut accorder un droit d'accès pour cette information à n'importe quel utilisateur"*

Propriétés de sécurité

- Si la politique d'autorisation est **cohérente**, alors il ne doit pas être possible, partant d'un état initial sûr (c'est-à-dire satisfaisant les propriétés de sécurité), d'atteindre un état d'insécurité (c'est-à-dire un état où les propriétés de sécurité ne sont pas satisfaites) en appliquant le schéma d'autorisation
- Les ITSEC définissent 3 propriétés de sécurité :
  - la **confidentialité** → prévention de la divulgation non autorisée de l'information
  - l'**intégrité** → prévention de la modification non autorisée de l'information
  - la **disponibilité** → prévention d'un refus d'accès à une ressource ou à une information normalement autorisée (déni de service)

Politique de sécurité → 3 directions

1. **physique** ⇒ ensemble de procédures et de moyens
   - protection des locaux et des biens contre des risques majeurs
     → *incendie, inondation,...*
   - contrôle des accès physiques aux matériels (info. & comm.)
     → *gardiens, codes, badges,...*
2. **administrative** ⇒ ensemble de procédures et moyens relatifs à la sécurité d'un point de vue organisationnel
   - structure de l'organigramme & répartition des tâches
     → *séparation des environnements de développement, d'industrialisation et de production des applicatifs*
   - propriétés de sécurité
     → *limiter les cumuls ou les délégations abusives de pouvoir*
     → *garantir une séparation des pouvoirs*

Politique de sécurité → 3 directions

3. **logique** ⇒ gestion du contrôle d'accès logique ⇒ repose sur un triple service d'identification, d'authentification et d'autorisation
   - elle spécifie qui a le droit d'accéder à quoi, et dans quelles circonstances

   ⇒ avant de se servir du système, tout utilisateur devra décliner son identité (**identification**) et prouver qu'il est bien la personne qu'il prétend être (**authentification**)

   - une fois la relation établie, les actions légitimes que peut faire cet utilisateur sont déterminées par la politique d'**autorisation**

Autorisation

⇒ Gérer et vérifier les droits d'accès aux ressources en fonction des règles spécifiées dans la politique de sécurité

- un **sujet** (entité qui demande l'accès → entité active) possède un droit d'accès sur un **objet** (entité à laquelle le sujet souhaite accéder → entité passive) si et seulement s'il est autorisé à effectuer la fonction d'accès (**action**) correspondante sur cet objet

- Exemple de mise en œuvre[3]
  - les droits d'accès peuvent être symboliquement représentés dans une matrice de droits d'accès dont les lignes représentent les sujets et les colonnes représentent les objets
  - une cellule de la matrice contient donc les droits d'accès d'un sujet sur un objet
  - la matrice est gérée conformément aux règles définies dans la politique de sécurité

3. cf. modèle HRU étudié plus loin

Règlement de sécurité

- Les définitions précédentes supposent que ce qui est permis est connu
  - le **règlement de sécurité** définit ce qui est autorisé et ce qui ne l'est pas (les règles de la politique de sécurité)
  - un règlement de sécurité est généralement un ensemble de :
    - → **permissions**
      - *les enfants ont la permission de minuit*
      - *tout médecin a le droit d'accéder aux dossiers médicaux de ses patients*
    - → **interdictions**
      - *les enfants ont la permission de minuit, sauf la petite sœur*
      - *les médecins n'ont pas le droit d'effacer des diagnostics déjà établis*
    - → **obligations**
      - *les médecins sont obligés de conserver les dossiers médicaux pendant la durée fixée par la loi*

Plan du cours

Critères d'évaluation

- Un système est sûr si et seulement si le règlement de sécurité ne peut pas être violé
- Les 1ers **critères d'évaluation de la sécurité** ont été définis aux États-Unis dans ce qui est couramment appelé le **Livre Orange** ou **TCSEC**[4]
  - → fondés à la fois sur des listes de fonctions de sécurité à remplir et sur les techniques employées pour la vérification, ces critères conduisent à classer les systèmes en sept catégories ou niveaux (D, C1, C2, B1, B2, B3, A1)

4. Trusted Computer System Evaluation Criteria

TCSEC

- Pour chaque niveau, quatre familles de critères sont définies :
  - la **politique d'autorisation** stipule une politique précise à suivre en fonction des différents niveaux de certifications visés
  - les **critères d'audit** précisent les fonctions requises en matière d'identification, d'authentification et d'audit des actions
  - les **critères d'assurance** fixent des recommandations concernant des méthodes de conception et de vérification utilisées afin d'augmenter la confiance de l'évaluateur
    - → il s'agit de garantir que le système implémente bien la fonctionnalité qu'il prétend avoir
  - les **critères de documentation** spécifient les documents qui doivent être fournis avec le produit lors de l'évaluation

TCSEC

- Caractéristiques principales des niveaux définis par les TCSEC :
  - → un système classé au niveau **D** est un système qui n'a pas été évalué
  - → jusqu'aux niveaux **C1** et **C2**, le système peut utiliser une politique discrétionnaire (cf. suite du cours)
  - → pour les niveaux **B1**, **B2**, et **B3** le système utilise une politique obligatoire (cf. suite du cours)
  - → un système classé **A1** est fonctionnellement équivalant à un système classé B3, sauf qu'il est caractérisé par l'utilisation de méthodes formelles de vérification pour prouver que les contrôles utilisés permettent bien d'assurer la protection des informations sensibles

TCSEC

- Les TCSEC visaient d'abord à satisfaire les besoins du DoD (*Department of Defense*) des États-Unis, privilégiant ainsi la confidentialité des données militaires
- Le manque de souplesse et la difficulté de leur mise en œuvre, ont conduit d'autres pays à élaborer et adopter leurs propres critères d'évaluation
  - ex-Communauté Européenne → ITSEC (1991)
  - Canada → CTCPEC (1993)
  - Japon → JCSEC (1992)

ITSEC

- Les **ITSEC**[5] sont le résultat d'harmonisation de travaux réalisés au sein de quatre pays européens : l'Allemagne, la France, les Pays-Bas et le Royaume-Uni
- La différence essentielle entre les TCSEC et les ITSEC réside dans la distinction entre fonctionnalité et assurance
  - → une **classe de fonctionnalité** décrit les fonctions que doit mettre en œuvre un système tandis qu'une **classe d'assurance** décrit l'ensemble des preuves qu'un système doit apporter pour montrer qu'il implémente les fonctions qu'il prétend fournir

5. Information Technology Security Evaluation Criteria

ITSEC

- Les ITSEC introduisent en plus la notion de **cible d'évaluation** (ou **TOE**[6])
- Une TOE rassemble les différents éléments liés au contexte d'évaluation
  - une politique de sécurité
  - une spécification des fonctions requises dédiées à la sécurité
  - une définition des mécanismes de sécurité (optionnelle)
  - la cotation annoncée de la résistance minimum des mécanismes
  - le niveau d'évaluation visé

6. Target Of Evaluation

ITSEC

- ITSEC ⇒ plusieurs classes de fonctionnalités de base :
  - → les classes de fonctionnalité **F-C1**, **F-C2**, **F-B1**, **F-B2**, **F-B3** sont des classes de confidentialité qui correspondent aux exigences de fonctionnalité des classes C1 à A1 dans les TCSEC
  - → la classe **F-IN** concerne les TOE pour lesquelles il existe des exigences d'intégrité (par exemple, pour les bases de données)
  - → la classe **F-AV** impose des exigences de disponibilité
  - → la classe **F-DI** impose des exigences élevées pour l'intégrité des données au cours de leur transmission
  - → la classe **F-DX** est destinée aux réseaux exigeants en matière de confidentialité et d'intégrité de l'information

Critères Communs

- Harmonisation des critères canadiens, européens et américains
  - ⇒ **critères communs** (en anglais *Common Criteria for Information Security Evaluation*)
  - ⇒ devenus maintenant une norme internationale (**ISO 15408**)
- Ces critères contiennent deux parties bien distinctes comme dans les ITSEC : fonctionnalité et assurance
- Les critères communs définissent également une cible d'évaluation (TOE) ainsi que les profils de protection

  NB : déjà existants dans les critères fédéraux américains (Federal Criteria, 1992)

Plan du cours

Modèles de sécurité

- Pour sécuriser un système informatique il est donc important de définir un **modèle de sécurité**
- Un tel modèle de sécurité exprime les **besoins de sécurité** du système d'information ; il inclut :
  - un **règlement de sécurité**
  - un **modèle d'administration** spécifiant qui a le droit de mettre à jour le règlement de sécurité
- Il existe différents modèles de sécurité
  - les modèles **discrétionnaires**
  - les modèles **obligatoires** (ou de **contrôle de flux**)
  - les modèles de **contrôle d'usage**

Modèles de sécurité

- Les modèles de contrôle d'accès discrétionnaires (ou **DAC** [7]) sont les plus répandus et implémentés mais sont vulnérables aux attaques par **cheval de Troie**
- Les modèles de contrôle d'accès obligatoires (ou **MAC** [8]) imposent des règles incontournables destinées à forcer le respect des exigences de sécurité
  - ils sont très formels, permettent d'atteindre un très haut niveau de sécurité (théoriquement) mais sont difficiles à implémenter
  - en général ils proposent des règlements qui visent à renforcer la propriété de confidentialité

7. Discretionary Access Control
8. Mandatory Access Control

Modèles de sécurité

- MAC ⇒ **contrôle de flux** : ces modèles proposent des solutions pour l'identification et l'élimination des canaux cachés
  - → *"un canal caché est un chemin de communication pouvant être exploité par un processus de transfert d'information de telle sorte qu'il contourne les mécanismes de contrôle d'accès, et qu'ainsi il viole la politique de sécurité"*
- Les modèles de contrôle d'usage sont plus récents et ont été proposés afin de prendre en compte les nouveaux besoins de sécurité présents dans certaines applications telles que la **gestion des droits numériques** (ou **DRM** [9])
  - → les DRM multimédia ont été très controversés
  - → mais les DRM peuvent répondre à de nouveaux besoins en entreprise (**E-DRM** [10])

9. Digital Rights Management
10. Enterprise-DRM

Plan du cours

Modèles de contrôle d'accès discrétionnaires

- Existent depuis les années 70 ; ont subi de nombreuses évolutions
  - un des premiers modèles proposés est celui de B. **Lampson** (structure de machine à états)

    "Protection", 5th Princeton Symposium on Information Sciences and Systems, 1971
  - progressivement amélioré → modèle **HRU** (M.A. Harrison, W.L. Ruzzo et J.D. Ullman)

    "Protection in Operating Systems", Communication of the ACM, 19(8), pp. 461-471, 1976
  - un des modèles les plus récents : **OrBAC** (Organization Based Access Control, 2003)

    "Organization Based Access Control", IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003), Lake Come, Italy, June 4-6, 2003

Modèles de contrôle d'accès discrétionnaires

- Quelques autres modèles de contrôle d'accès :

RBAC Role Based Access Control
TBAC Task Based Access Control
VBAC View Based Access Control
TMAC TeaM based Access Control
...

Modèles de contrôle d'accès discrétionnaires

HRU

- Le modèle HRU aurait pu s'appeler IBAC (Identity Based Access Control) car il repose sur l'identité des entités actives du système d'information
- Il introduit les concepts de **sujet**, **action** et **objet**

sujet : c'est l'entité active ; il désigne un utilisateur, le système lui même, un processus s'exécutant pour le compte d'un utilisateur ou un processus système

objet : c'est l'entité passive ; il désigne une information ou une ressource à laquelle un sujet peut accéder pour réaliser une action

action : désigne l'effet recherché lorsqu'un sujet accède à un objet (ex : lire, écrire)

Modèles de contrôle d'accès discrétionnaires

HRU

- L'objectif du modèle HRU est de contrôler tout accès direct des sujets aux objets via l'utilisation des actions
- Le règlement (politique d'autorisations) correspond à un ensemble d'autorisations positives (permissions) du type :
  - → *"le sujet **s** a la permission de réaliser l'action **a** sur l'objet **o**"*
- La **politique d'autorisation par défaut** est **fermée**
  - ⇒ par défaut tous les accès sont interdits
  - → *"tout ce qui n'est pas explicitement autorisé est interdit"*

Modèles de contrôle d'accès discrétionnaires

HRU

- Le règlement est formalisé à l'aide d'une **matrice de contrôle d'accès**
  - les lignes représentent les sujets
  - les colonnes représentent les objets
  - une cellule (intersection d'une ligne et d'une colonne) contient l'ensemble des actions qu'un sujet a la permission d'effectuer sur un objet
- Le modèle HRU a été implanté dans la plupart des systèmes d'exploitation actuels tels que Windows ou Unix

Modèles de contrôle d'accès discrétionnaires

HRU

- La matrice n'est pas directement implanté ; il existe en fait deux approches selon que l'implantation repose sur une décomposition en colonnes ou en ligne de la matrice :
  - la décomposition en colonnes consiste à associer à chaque objet un descripteur appelé **liste de contrôle d'accès** (ou **ACL** [11])
    - → une ACL représente l'ensemble des sujets ayant des droits d'accès sur l'objet considéré avec pour chaque sujet l'ensemble des actions que ce sujet peut réaliser sur l'objet
  - la décomposition en lignes consiste à associer à chaque sujet une **liste de capacités**
    - → un ensemble de capacités associé à un sujet représente l'ensemble des objets auxquels le sujet considéré a accès avec pour chaque objet la liste des actions que peut réaliser le sujet

11. Access Control List

Modèles de contrôle d'accès discrétionnaires

HRU

	Sam	Joe	Code	Data
Sam			read,write,execute	read,write
Joe			read,execute	read

- ACL
  - → ⟨Code,(Sam,(r,w,x)),(Joe,(r,x))⟩
  - → ⟨Data,(Sam,(r,w)),(Joe,(r))⟩
- Capacités
  - → ⟨Sam,(Code,(r,w,x)),(Data,(r,w))⟩
  - → ⟨Joe,(Code,(r,x)),(Data,(r))⟩

Modèles de contrôle d'accès discrétionnaires

HRU

- Notion de **propriétaire**
  - le propriétaire d'un objet est celui qui a créé l'objet
  - le propriétaire d'un objet dispose de tous les droits sur l'objet
  - le propriétaire d'un objet peut **déléguer** à un autre sujet les droits sur son objet

Modèles de contrôle d'accès discrétionnaires

HRU

- Le modèle d'administration du modèle HRU consiste en un ensemble de **règles** définissant dans quelles conditions la matrice peut être modifiée ; ces règles utilisent les **primitives** suivantes :
  - donner un droit **r** à un sujet **s** sur un objet **o**
  - créer un sujet **s**
    - → *ajouter une ligne et une colonne car le sujet est aussi un objet*
  - créer un objet **o**
    - → *ajouter une colonne*
  - enlever un droit **r** à un sujet **s** sur un objet **o**
  - détruire un sujet **s**
  - détruire un objet **o**

Modèles de contrôle d'accès discrétionnaires

HRU

- Format général d'une règle HRU

```
Command α(x_1, x_2, ..., x_k)
    If r_1 ∈ M(s_1, o_1) and r_2 ∈ M(s_2, o_2) and ... and r_m ∈ M(s_m, o_m)
    Then
        /* op_i = primitive */
        op_1 ; op_2 ; ... ; op_n
End
```

- Exemples :
  - Command CREATE(*user* , *file*)
    create object *file* ;
    enter *owner* into $M(user, file)$;
  - Command CONFER_r(*user* , *friend* , *file*)
    If $owner \in M(user, file)$
    Then enter *r* into $M(friend, file)$;

Modèles de contrôle d'accès discrétionnaires

HRU

- Le modèle HRU a été fréquemment implémenté

Unix
- objets = fichiers/répertoires
- actions = {r,w,x}
- permissions représentées sous forme d'ACL
- administration → commande chmod

SQL
- objets = tables/vues
- actions = {select,update,delete,insert}
- administration → commandes grant et revoke

Modèles de contrôle d'accès discrétionnaires

HRU

- Avantages du modèle HRU
  - simple, souvent implémenté
    - → Unix, Windows, SQL,...
  - administration décentralisée de la règlementation
- Limites du modèle HRU : règlement complexe à exprimer et à administrer
  - énumération des autorisations $\langle sujet, action, objet \rangle$
    - → fastidieux, coûteux en mémoire
  - ⇒ constitution de **groupes** d'utilisateurs pour réduire la taille de la matrice
    - → maintenance des groupes délicate car un sujet peut appartenir à plusieurs groupes
  - mise à jour du règlement à chaque création de sujet et d'objet

Modèles de contrôle d'accès discrétionnaires

HRU

- Limites du modèle HRU
  - problème de la fuite des droits (*safety*)
    - → considérant un état de la matrice, il est impossible de s'assurer qu'un sujet ne pourra jamais recevoir un droit particulier sur un certain objet
  - vulnérable aux attaques par cheval de Troie
    - → le modèle HRU est vulnérable aux attaques par cheval de Troie effectuant des recopies de fichiers (cf. sécurité multi-niveaux)
    - → défaut aggravé par le fait que les systèmes informatiques sont maintenant tous interconnectés (Internet)

Modèles de contrôle d'accès discrétionnaires

RBAC

- Le modèle RBAC[12] propose de structurer le règlement autour du concept de **rôle**
  → un rôle est un concept organisationnel **structurant les sujets**
    - des rôles sont affectés aux utilisateurs conformément à la fonction que ces utilisateurs jouent dans l'organisation
    - les autorisations (droits d'effectuer des actions sur des objets) sont affectées aux rôles
- Le modèle RBAC ne considère que des autorisations positives et suppose une politique par défaut fermée

12. Sandhu (1996)

Modèles de contrôle d'accès discrétionnaires

RBAC

- Le modèle RBAC introduit la notion de **session**
  - tout utilisateur doit initier une session avant de pouvoir accéder aux objets
  - dans le cadre de cette session il peut activer un ou plusieurs rôles parmi tous les rôles qui lui ont été attribués
    - → les droits (privilèges) de l'utilisateur seront alors les droits appartenant au(x) rôle(s) activé(s)
- Les rôles peuvent être organisés **hiérarchiquement**
  - ⇒ les rôles **héritent** des autorisations des rôles hiérarchiquement inférieurs
    Ex : "cardiologue" et "radiologue" héritent de "médecin"

Modèles de contrôle d'accès discrétionnaires

RBAC

- Le modèle RBAC introduit la notion de **contrainte** permettant de spécifier des réglementation de type **séparation de tâches**
  - une séparation de tâches **statique** prévoit que 2 rôles (par exemple médecin et infirmier) ne peuvent pas être assignés à un même utilisateur
  - une séparation de tâches **dynamique** prévoit que 2 rôles (par exemple médecin libéral et chirurgien) ne peuvent être activés en même temps par un même utilisateur

Modèles de contrôle d'accès discrétionnaires

RBAC

- Le modèle RBAC initial ne définissait pas de modèle d'administration
  → en particulier, il ne prévoyait pas qui avait le droit de créer et mettre à jour les rôles
- Pour combler cette lacune, le modèle ARBAC (Administrative Role Based Access Control) a été proposé
  - le modèle ARBAC est un modèle d'administration pour le modèle RBAC
  - il est lui aussi basé sur les rôles

Modèles de contrôle d'accès discrétionnaires

RBAC

- Avantages du modèle RBAC
  - structuration du règlement de sécurité
  - de plus en plus implémenté
    - → versions récentes de SQL
    - → Unix Solaris v8
    - → API Authorization Manager RBAC de Windows Server 2003
- Inconvénients
  - ce modèle est toujours vulnérable aux attaques par cheval de Troie
  - il nécessite de mettre en place une procédure d'administration des rôles

Modèles de contrôle d'accès discrétionnaires

- Les systèmes d'information actuels devenant de plus en plus sophistiqués, de nouveaux modèles de sécurité sont apparus
- Les nouveaux modèles dérivés de HRU et RBAC introduisent de nouvelles possibilités
  - → workflows, vues, contextes,...

Modèles de contrôle d'accès discrétionnaires

TBAC

- Pour répondre au besoin de contrôler des actions composites (dans une application de type workflow), le modèle TBAC introduit la notion de **tâche**
  - une tâche est une suite d'actions élémentaires
    - → dans une agence de voyage, achat d'un billet = réservation du billet + paiement + edition de la facture
    - → les autorisations sont définies sur les tâches
    - → il reste toutefois possible de contrôler les actions élémentaires
  - la notion d'**autorisation *just in time*** est aussi introduite
    - → la permission d'éditer une facture ne doit être activée qu'après réservation et paiement

Modèles de contrôle d'accès discrétionnaires

VBAC

- Le modèle de sécurité de SQL introduit la notion de **vue**
  - Une vue permet de structurer les objets (les tuples)
    - → une vue est le résultat d'une requête auquel on a donné un nom
  - les autorisations sont définies sur les vues
    - → le modèle SQL peut donc être qualifié de VBAC (View Based Access Control)
- Le règlement est défini à l'aide des commandes `grant` et `revoke` qui permettent respectivement d'accorder ou de supprimer une permission à un utilisateur
- La norme SQL/3 inclut le concept de rôle
  - → on pourrait donc désigner le modèle de sécurité de cette norme comme étant de type VRBAC

Modèles de contrôle d'accès discrétionnaires

VBAC

- La notion de vue pourrait aussi concerner les systèmes de fichiers des systèmes d'exploitation
  - actuellement, les systèmes d'exploitation existants ne proposent que la notion physique de répertoire pour structurer les objets (fichiers)
  - avec un langage de requête approprié il pourrait être intéressant de définir une vue résultat de la requête renvoyant la liste des fichiers .doc du répertoire de Pierre et d'utiliser cette vue dans la définition du règlement de sécurité
    - → cette approche est utilisée par des modèles récents de contrôle d'accès pour données XML, le langage d'interrogation utilisé étant XPath

Modèles de contrôle d'accès discrétionnaires

Contextes

- En pratique, de nombreuses autorisations ne sont pas statiques mais dépendent de conditions qui, si elles sont satisfaites, permettent d'activer dynamiquement les autorisations

⇒ On parle d'**autorisations contextuelles**
  - contexte **temporel** → permission pendant les heures de travail
  - contexte **géographique** → permission uniquement à l'intérieur de l'enceinte sécurisée
  - contexte **provisionnel** → permission si d'autres actions ont été réalisées comme dans le cas d'un workflow

- Pour prendre en compte ces besoins, différents modèles à base de règles ont été définis (modèles de type Rule-BAC)
  - un règlement correspond alors à un ensemble de règles du type *condition* → *permission*

Modèles de contrôle d'accès discrétionnaires

Interdictions

- De nouveaux modèles de contrôle d'accès permettent d'exprimer des autorisations négatives (**interdictions**) sont apparus
- Utiliser des interdictions peut souvent répondre à un besoin
  - certaines règlementations sont plus faciles à exprimer à l'aide d'interdictions
    → vidéo interdite au moins de 18 ans
  - combiner des permissions et des interdictions permet de spécifier de manière concise des autorisations souffrant d'**exceptions**
    1. les infirmiers ont l'interdiction d'accéder au dossier médical des patients
    2. en situation d'urgence, les infirmiers ont la permission d'accéder au dossier médical du patient

Modèles de contrôle d'accès discrétionnaires

Interdictions

- Utiliser simultanément des permissions et des interdictions peut créer des **conflits** dans la règlementation
  - → dans l'exemple précédent, la règle 1 est en conflit avec la règle 2 dès lors que nous sommes dans un contexte d'urgence
- Pour résoudre ces conflits diverses approches ont été proposées :
  - les interdictions (ou les permissions) l'emportent toujours
  - les règles reçoivent des niveaux de priorité
  - le plus spécifique l'emporte
  - ordre dans lequel les règles sont écrites (*first-matching applies*)

Modèles de contrôle d'accès discrétionnaires

OrBAC

- Actuellement, il n'existe pas de modèle de contrôle d'accès permettant de répondre à tous les besoins de sécurité énoncés
- Néanmoins le modèle OrBAC[13] est certainement un des modèles de contrôle d'accès les plus complets

http://orbac.org

13. Organization Based Access Control (2003)

Plan du cours

Modèles de contrôle d'accès obligatoires

- Depuis 1975 on sait que les modèles de contrôle d'accès ne permettent pas de prendre en compte les applications piégées par un cheval de Troie (opérant par recopie de fichiers)
- Afin de prendre en compte cette possibilité, des modèles dits de contrôle des flux ont été définis parallèlement à la définition des modèles de contrôle d'accès
- Le premier modèle de contrôle des flux est le modèle de **Bell & LaPadula** (1975) (cf. sécurité multi-niveaux)
- D'autres modèles plus sûrs que le modèle de Bell & LaPadula ou répondant à des objectifs différents ont été proposés depuis
  → *non Interférence, Bell & LaPadula étendu, causalité, Biba,...*

Modèles de contrôle d'accès obligatoires

- Appelés modèles **obligatoires** car le règlement de sécurité est simple et s'impose à tous les utilisateurs (il ne contient pas de règle adressant un utilisateur en particulier)
- Pour comprendre l'intérêt des modèles de contrôle des flux, il est nécessaire de revenir à la notion de sujet :
  - → *un sujet est un **utilisateur** ou un **processus** s'exécutant pour le compte d'un utilisateur*

Modèles de contrôle d'accès obligatoires

- Tous les modèles de sécurité font implicitement les hypothèses suivantes sur les sujets :
  - un utilisateur peut potentiellement chercher à violer le règlement en tentant d'accéder à des objets pour lesquels il n'a pas d'autorisation
  - un utilisateur est supposé **de confiance**
    - ⇒ un utilisateur ne va pas délibérément divulguer de l'information à laquelle il a légalement accès
  - la plupart des processus ne sont pas de confiance car ils peuvent être potentiellement piégés et contenir un cheval de Troie
    - pb : un processus hérite des droits de l'utilisateur pour le compte duquel il s'exécute...
    - les seuls processus supposés de confiance sont ceux implantant les mécanismes de sécurité (les contrôles d'accès par exemple)

Modèles de contrôle d'accès obligatoires

Cheval de Troie

- Considérons un médecin qui utiliserait une application médicale piégée
  - ⇒ le cheval de Troie pourrait, **à l'insu du médecin**, transmettre le contenu d'un dossier médical par Internet à une personne non autorisée
- Les modèles de contrôle d'accès ne permettent pas d'empêcher de telles actions malveillantes
  - en effet le piège introduit dans l'application médical ne viole pas la réglementation de sécurité
    - le médecin (et donc l'application médicale) a le droit d'accéder à un dossier médical
    - le médecin (et donc l'application médicale) a le droit d'accéder à Internet (à un dictionnaire médical en ligne par exemple)

Modèles de contrôle d'accès obligatoires

Sécurité multi-niveaux

- Objectif de sécurité : **confidentialité**
- Système informatique représenté par le modèle Sujet-Objet
  - chaque sujet reçoit un niveau d 'habilitation
    - → Public, Confidentiel, Secret,...
  - chaque objet reçoit un niveau de classification
    - → Public, Confidentiel, Secret,...
- Règlement de sécurité (1 phrase)
  - un sujet s est autorisé à **connaître** la valeur de l'objet o si et seulement si $hab(s) \geq class(o)$
    - ⇒ un utilisateur **habilité** C aura le droit de connaître une information **classifiée** P ou C mais n'aura pas le droit de connaître une information **classifiée** S

Modèles de contrôle d'accès obligatoires

Sécurité multi-niveaux

- Modèle de Bell & LaPadula ⇒ les deux propriétés suivantes sont **nécessaires** (mais pas suffisantes) pour garantir le règlement de sécurité :

No Read-up un sujet s ne peut **lire** le contenu d'un objet o que si $hab(s) \geq class(o)$

⇒ un utilisateur habilité C a le droit de lire une information classifiée P ou C mais n'a pas le droit de lire une information classifiée S

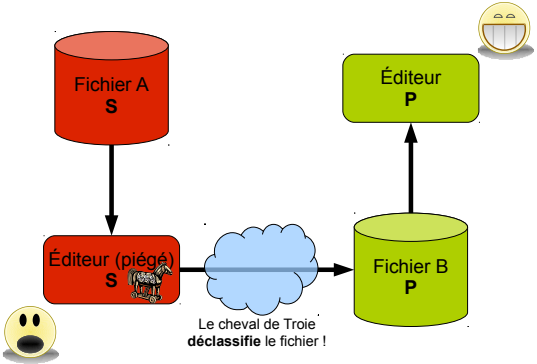
No Write-down un sujet s ne peut **modifier** le contenu d'un objet o que si $hab(s) \leq class(o)$

⇒ un utilisateur habilité C a le droit de modifier une information classifiée C ou S mais n'a pas le droit de modifier une information classifiée P

Modèles de contrôle d'accès obligatoires

Sécurité multi-niveaux

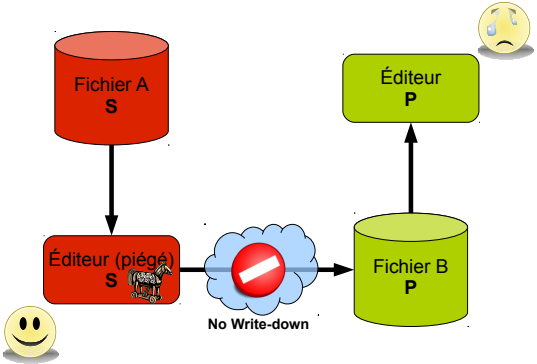
- Cheval de Troie



Modèles de contrôle d'accès obligatoires

Sécurité multi-niveaux

- Avec le modèle Bell & LaPadula



Modèles de contrôle d'accès obligatoires

Sécurité multi-niveaux

- Un utilisateur **habilité** à un niveau l peut initier une session à tout niveau **dominé** par le niveau l

  ex : un utilisateur habilité `Secret` se connecte au niveau `Secret` pour accéder à des informations secrètes mais se connecte au niveau `Public` pour modifier des informations publiques

  ⇒ permet de **déclassifier** certaines informations

Modèles de contrôle d'accès obligatoires

- Avantages des modèles de contrôle de flux :
  - permettent de lutter contre les chevaux de Troie opérant par recopie de fichiers
  - ⇒ procurent un niveau de sécurité supérieur aux modèles de contrôle d'accès
- Inconvénients :
  - le règlement est très rigide
  - ⇒ modèles généralement réservés à des usages **militaires**
  - implémentation plus complexe

Autres modèles

- Les deux conditions de modèle de Bell & LaPadula sont **nécessaires** mais **pas suffisantes**
  - les modèles de Non Interférence et de Causalité prennent en compte beaucoup plus de pièges que le modèle de Bell & LaPadula mais ils sont pratiquement impossibles à implémenter
  - le modèle de Biba est l'équivalent du modèle de Bell & LaPadula pour l'intégrité

⇒ Il est nécessaire de contrôler **tous** les flux d'information :
  - contrôle des canaux cachés
  - contrôle de l'inférence

Contrôle des canaux cachés

- Un canal caché est un **canal de communication non prévu** mais pouvant potentiellement servir à communiquer de l'information de façon illégale
- Les canaux cachés résultent très souvent de particularités liées à l'**implémentation**

  ⇒ il est difficile d'en tenir compte dans un modèle théorique représentant un certain niveau d'abstraction

Contrôle des canaux cachés

- Exemples de canaux cachés :
  - un processus piégé exécuté au niveau S peut accéder à une ressource quelconque pour transmettre des infos sensibles ; un sujet P peut observer ces accès et en déduire ces infos sensibles
  - un processus piégé exécuté au niveau S peut moduler son temps d'exécution pour transmettre des infos sensibles $\rightarrow$ canal caché temporel
  - on peut observer la consommation d'énergie d'un processus pour en déduire des infos sensibles

Contrôle des canaux cachés

Ils espionnent un ordinateur grâce aux ondes générées par...l'USB

- Fin 2013 → Edward Snowden → NSA → **COTTONMOUTH**
  - Article 01net du 01/09/2016
  - Un petit mouchard matériel qui vient se loger dans une prise USB et qui permet d'exfiltrer des données par ondes radio d'un ordinateur qui n'est pas connecté sur un réseau informatique
  - Alternative USBee : Le bus de données en USB s'appuie sur deux fils électriques dont la tension peut s'inverser. Si la tension s'inverse, alors c'est un «0», sinon c'est un «1».
  - Les chercheurs ont alors eu l'idée de n'envoyer que des zéros vers le matériel USB → la succession de changements de tension crée une belle onde électromagnétique.

Contrôle des canaux cachés

Ils espionnent un ordinateur grâce aux ondes générées par...l'USB (suite)

- Moduler la fréquence avec des zéros
  - Les chercheurs ont alors développé un algorithme qui permet de moduler la fréquence de cette onde en jouant sur la longueur des séries de «0» envoyées.
  - Répéter par exemple un bloc constitué d'une douzaine de «0» et d'une douzaine de «1» (111111111111000000000000) donnera une fréquence différente qu'une banale succession de «0» (000000000000000000000000).
  - La modulation de la fréquence permet ensuite de coder le message à exfiltrer. Pour capter le message, il suffit d'avoir une antenne et un logiciel d'analyse spectral tel que GNU Radio. Ce qui représente un investissement d'une trentaine de dollars.

Contrôle des canaux cachés

Ils espionnent un ordinateur grâce aux ondes générées par...l'USB (fin)

- Débit & utilisation
  - Les tests effectués ont permis de faire passer des messages d'une pièce à l'autre avec un débit compris entre 20 et 80 octets par seconde → regarder un film en HD est donc totalement hors de portée, mais c'est suffisant pour envoyer une clé de chiffrement en peu de temps.
  - COTTONMOUTH n'est pas encore à mettre au placard pour autant, car ce mouchard de la NSA est capable non seulement d'émettre, mais aussi de recevoir des données. Ce qui n'est pas le cas d'USBee → 2ème version ?

Contrôle de l'inférence

- Un utilisateur peut **déduire** des informations sensibles en utilisant des informations qu'il est autorisé à connaître

  ex : un médecin est tenu au secret médical

  pourtant il délivre une ordonnance qui peut être aisément lue par des tiers (pharmacien, famille, clients dans la pharmacie,...)

  → la lecture de cette ordonnance peut révéler, par déduction, la nature de la maladie du patient

⇒ Problème difficile à résoudre complètement dans la mesure où il est difficile de recenser toutes les connaissances possédées par l'utilisateur

Contrôle de l'inférence

- Pour déduire (ou inférer) des informations sensibles un utilisateur peut utiliser :
  - des informations présentes dans le système informatique et auxquelles il a légalement accès
    - ⇒ contrôle de l'inférence possible
  - des connaissances générales non représentées dans le système informatique
    - ⇒ contrôle de l'inférence difficile

Plan du cours

Modèles de contrôle d'usage

- L'objectif du contrôle d'usage est de contrôler non seulement l'accès au document mais également l'**usage qui en est fait**
  - ex : initialement, le contrôle d'usage visait principalement (mais pas seulement) à contrôler la recopie des fichiers
- Idée [14] : contrôle d'usage $\equiv$ obligations

14. dixit F. & N. Cuppens

Modèles de contrôle d'usage

- Les modèles de contrôle d'usage dont la première version est le modèle **UCON**[15] permettent d'énoncer des règles de sécurité qu'il est difficile d'implanter avec des mécanismes classiques de contrôle d'accès :
  - → l'acheteur de ce morceau de musique ne pourra l'écouter que 10 fois au plus
  - → l'utilisateur de ce document ne pourra effectuer qu'une seule copie de sauvegarde
  - → le médecin aura l'obligation de mettre à jour le dossier médical du patient avant de pouvoir imprimer l'ordonnance

15. Park, Sandhu (2004)

Modèles de contrôle d'usage

- La mise en œuvre d'un règlement de contrôle d'usage se fait généralement en utilisant des techniques de DRM

  NB : les DRM se caractérisent par le fait que les contrôles de sécurité s'effectuent non pas du coté du serveur mais **du coté du client**

- La partie du logiciel client qui effectue les contrôles de sécurité (noyau de sécurité) **doit être de confiance**

  → **par définition**, le noyau de sécurité ne peut être contourné et est dépourvu de failles, vulnérabilités, cheval de Troie,...

Modèles de contrôle d'usage

- Applications des DRM :
  - Initialement orientés vers la protection des droits d'auteurs et des intérêts commerciaux des distributeurs de contenus multimédia (films, musique,...)
  - Maintenant, les DRM sont de plus en plus utilisés dans des applications dont l'objectif est de contrôler la distribution de contenus sensibles (*Entreprise-DRM*)

    ex : FLUOR, projet ANR-SESUR (2008-2011)
    *convergence du contrôle de **FL**ux et d'**U**sage dans les **OR**ganisations*

Modèles de contrôle d'usage

OrBAC

- Retour sur le modèle OrBAC :
  - → contrôle d'accès
  - → contrôle d'usage
- La politique de sécurité permet :
  - permissions
  - interdictions
  - obligations
  - règles contextuelles

http://orbac.org

Plan du cours

Mise en œuvre des modèles

- Approche dite du **noyau de sécurité** (ou TCB[16])
  NB : côté serveur ou côté client (cf. DRM)
- Le noyau est supposé fiable (*trusted*), i.e. dépourvu de :
  - failles
  - vulnérabilités
  - pièges
  - ...

⇒ Idéalement il doit donc être le plus petit possible

16. Trusted Computing Base

Mise en œuvre des modèles

- Les fonctions du noyau de sécurité sont :
  - → authentification des utilisateurs
  - → contrôle des accès (ACL, No Read-up, No Write-down,...)
  - → chiffrement & déchiffrement de données
  - → ...

⇒ Ces mécanismes de sécurité doivent garantir le règlement de sécurité (***policy enforcement***)

Plan du cours

Rappel

- Selon les ITSEC (critères européens), la sécurité informatique recouvre 3 objectifs de sécurité :
  - la **confidentialité** → prévention de la divulgation non autorisée de l'information
  - l'**intégrité** → prévention de la modification non autorisée de l'information
  - la **disponibilité** → prévention d'un refus d'accès à une ressource ou à une information normalement autorisée (déni de service)
- Une violation de sécurité peut être accidentelle ou résulter d'une action malveillante

  ⇒ un SGBD procure certains mécanismes permettant d'assurer un certain degré de sécurité

SGBD & mécanismes de sécurité

- Pour assurer la **confidentialité** des informations
  - cryptographie
  - contrôle d'accès & assignation des droits (privilèges) (cf. `GRANT` et `REVOKE`)
  - définition de rôles
  - utilisation de vues
- Pour assurer l'**intégrité** des informations
  - contraintes d'intégrité & mécanismes associés
  - gestion des transactions (concurrence d'accès)
  - contrôle d'accès & assignation des droits (privilèges) (cf. `GRANT` et `REVOKE`)
  - reprise après panne

Plan du cours

Privilèges & vues

- Dans cette partie du cours nous nous intéresserons uniquement aux mécanismes suivants :
  - assignation de droits (commandes SQL `GRANT` et `REVOKE`)
  - définition de rôles
  - utilisation de vues

- Dans un SGBDR standard, le mécanisme d'assignation de droits permet au DBA d'accorder des privilèges à des utilisateurs
  - privilèges systèmes
  - privilèges sur des objets

Privilèges systèmes

- Quelques privilèges systèmes :
  - `CREATE TABLE`
  - `CREATE USER`
  - `CREATE VIEW`
  - `SELECT ANY TABLE`
  - `ADMIN OPTION FOR` ⟨*system_right*⟩
    privilège permettant d'octroyer/retirer des privilèges systèmes

Privilèges sur des objets

- Quelques objets :
  - tables
  - vues
  - index
- Quelques privilèges sur des objets :
  - `DELETE FROM` ⟨*table*⟩
  - `INSERT INTO` ⟨*table*⟩
  - `SELECT FROM` ⟨*table* | *view*⟩
  - `UPDATE` ⟨*table*⟩
  - `GRANT OPTION FOR` ⟨*object_right*⟩
    droit d'octroyer/retirer des privilèges sur des objets

Oracle

Privilèges systèmes

NB : Les règles suivantes sont basées sur le système d'assignation des privilèges d'Oracle

- Qui peut octroyer un privilège système ?
  - le DBA qui possède tous les privilèges
  - un utilisateur détenteur d'un privilège système avec l'`ADMIN OPTION` peut octroyer ce privilège
- Qui peut révoquer un privilège système ?
  - le DBA
  - un utilisateur détenteur d'un privilège système avec l'`ADMIN OPTION` peut retirer ce privilège à un utilisateur descendant dans le graphe d'autorisation de ce privilège

Oracle

Privilèges systèmes

- Exemple :
  - DBA octroie P_1 (with `ADMIN`) à $user_1$
  - $user_1$ octroie P_1 à $user_2$ (with `ADMIN`)
  - $user_2$ octroie P_1 à $user_3$
  - $user_2$ peut révoquer P_1 de $user_3$ mais pas de $user_1$
  - $user_1$ peut révoquer P_1 de $user_2$ ou $user_3$ mais pas de DBA !

$$DBA \xrightarrow[\texttt{ADMIN}]{P_1} user_1 \xrightarrow[\texttt{ADMIN}]{P_1} user_2 \xrightarrow{P_1} user_3$$

Oracle

Privilèges objets

- Qui peut octroyer un privilège objet ?
  - le propriétaire (i.e. le créateur de l'objet) qui possède tous les privilèges sur l'objet
  - un utilisateur détenteur d'un privilège objet avec la `GRANT OPTION`
- Qui peut révoquer un privilège objet ?
  - le DBA
  - le détenteur original du privilège uniquement

Oracle

Révocation en cascade

- Il n'y a pas d'effet en cascade pour la révocation d'un privilège système, que ce privilège ait été octroyé avec l'`ADMIN OPTION` ou non
- Exemple :
  - → DBA octroie `CREATE TABLE` à $user_1$
  - → $user_1$ octroie `CREATE TABLE` à $user_2$
  - → DBA révoque `CREATE TABLE` de $user_1$
  - → $user_2$ **conserve** `CREATE TABLE`

Oracle

Révocation en cascade

- Il y a effet en cascade pour la révocation d'un privilège objet, que ce privilège ait été octroyé avec la `GRANT OPTION` ou non
- Exemple :
  - → DBA octroie P_1 (with `GRANT`) à $user_1$
  - → $user_1$ octroie P_1 à $user_2$ (with `GRANT`)
  - → $user_2$ octroie P_1 à $user_3$
  - → si $user_1$ révoque P_1 de $user_2$ alors P_1 **est aussi révoqué** de $user_3$

Oracle

Révocation

NB : Les privilèges `ADMIN OPTION` et `GRANT OPTION` ne peuvent être retirés de façon sélective

- Exemple :
  - → DBA octroie P_1 (with `ADMIN`) à $user_1$
  - → pour retirer l'`ADMIN OPTION` de $user_1$ DBA doit d'abord retirer P_1 de $user_1$ puis octroyer de nouveau P_1 à $user_1$

Oracle

Commandes SQL

- Octroyer/révoquer des privilèges systèmes

GRANT <right(s)> **TO** <user(s)> | PUBLIC [WITH ADMIN OPTION]
REVOKE <right(s)> **FROM** <user(s)> | PUBLIC

- Octroyer/révoquer des privilèges objets

GRANT ALL | <right(s)> **ON** <object> **TO** <user(s)> | PUBLIC [WITH GRANT OPTION]
REVOKE <right(s)> **ON** <object> **FROM** <user(s)> | PUBLIC

Oracle

Rôles

- Afin de simplifier l'administration des privilèges il est possible de définir des **rôles**
- Commandes SQL sur les rôles :
  - création d'un rôle

    ```
    CREATE ROLE <role>
    ```

  - assignation de privilèges aux rôles : les commandes `GRANT` et `REVOKE` vues précédemment peuvent prendre un rôle en paramètre à la place du nom d'un utilisateur
  - affectation de rôles aux utilisateurs (ou à d'autres rôles)

    ```
    GRANT <role> TO <user(s)|role(s)>
    ```

Oracle

Rôles

- Activation/désactivation de rôles
  - la commande "SET ROLE <role>" permet d'activer ou de désactiver un rôle pour la session courante
  - lorsqu'un utilisateur ouvre une session sous Oracle, tous les rôles par défaut sont activés ; les autres rôles doivent être activés via la commande SET ROLE
  - la commande "SET ROLE NONE" désactive tous les rôles pour la session courante (y compris tous les rôles par défaut)

Oracle

Granularité des privilèges

- Le mécanisme de contrôle d'accès par défaut pour Oracle SQL est basé sur la notion d'**objet Oracle**
  - ⇒ si un utilisateur dispose du privilège "SELECT ON Employes" alors il lui est possible de voir **toutes les lignes** de la table "Employes"

⇒ Pour restreindre l'accès à certaines lignes et/ou masquer certaines colonnes
  - → utilisation de la notion de **vues**

Oracle

Vues

- Une vue est une table virtuelle résultat d'une requête
- Une vue peut être utilisée dans des requêtes au même titre qu'une table permanente
- Par contre, la mise à jour au travers d'une vue est soumise à des restrictions
- Commandes SQL pour créer/supprimer une vue :

```
CREATE VIEW <name> [(column_name_list)] AS <request>
DROP VIEW <name>
```

Oracle

Vues

- Requêtes imbriquées dans la clause `from`

Total des soldes et des emprunts pour chaque client

```
select temp1.nom,tot1,tot2
from ( select nom,sum(solde) tot1 from compte group by nom
       union
       (select nom,0 from client
        where nom not in (select nom from compte))
     ) as temp1,
     ( select nom,sum(montant) tot2 from emprunt group by nom
       union
       (select nom,0 from client
        where nom not in (select nom from emprunt))
     ) as temp2
where temp1.nom = temp2.nom;
```

Plan du cours

Exemples

Réglementation médicale (© Alban Gabillon / UPF)

- Table `Patient`

Login	Nom	Prenom	Age	Sexe
pfranck	Franck	Patricia	60	F
mrobert	Robert	Martin	35	M

- Table `Diagnostic`

Login	Maladie
pfranck	ulcere
mrobert	pneumonie
mrobert	asthme

Exemples

Réglementation médicale (© Alban Gabillon / UPF)

- Règlement de sécurité :
  - les médecins ont la permission de consulter et modifier les deux tables sans restriction
  - les secrétaires ont la permission de voir la table patient
  - les infirmières ont la permission de voir les deux tables
  - les patients ont la permission de voir les informations qui les concernent

Exemples

Réglementation médicale (© Alban Gabillon / UPF)

Sujets
- les médecins
- les infirmières
- les secrétaires
- les patients

Objets
- les 2 tables
- les tuples de chaque table

Actions
- `SELECT`
- `INSERT`
- `UPDATE`
- `DELETE`

Exemples

Réglementation médicale (© Alban Gabillon / UPF)

- Création des rôles :

```
CREATE ROLE Medecin;
CREATE ROLE Infirmiere;
CREATE ROLE Secretaire;
CREATE ROLE Malade;
```

- Assignation des droits à chaque rôle :

```
GRANT ALL ON Patient TO Medecin;
GRANT ALL ON Diagnostic TO Medecin;
GRANT SELECT ON Patient TO Infirmiere;
GRANT SELECT ON Diagnostic TO Infirmiere;
GRANT SELECT ON Patient TO Secretaire;
```

  ⇒ pb pour la dernière règle car le niveau de granularité du modèle de sécurité de SQL est la table et non le tuple

Exemples

Réglementation médicale (© Alban Gabillon / UPF)

- Solution : créer 2 vues

```
CREATE VIEW VPatient AS
    SELECT * FROM Patient where Login=user;

CREATE VIEW VDiagnostic AS
    SELECT * FROM Diagnostic where Login=user;

GRANT SELECT ON VPatient to Malade;
GRANT SELECT ON VDiagnostic to Malade;
```

Exemples

Réglementation médicale (© Alban Gabillon / UPF)

- Affectation des rôles aux utilisateurs :

```
GRANT Medecin TO ...
GRANT Infirmiere TO ...
GRANT Secretaire TO ...
GRANT Malade TO pfranck;
GRANT Malade TO mrobert;
```

Exemples

Réglementation obligatoire (© Alban Gabillon / UPF)

- Implémentation d'un règlement multi-niveaux
  - table multi-niveaux `AIRCRAFT`

Name	Type	Speed	Range	Nuclear_bomb
F16	Supersonic	Mach 2.05	2000 km	NO
Mirage	Supersonic	Mach 2.2	2000 km	NO
Rafale	Supersonic	Mach 1.8	4000 km	YES
Firefox	Hypersonic	Mach 6	6000 km	YES

  - niveau de granularité = valeur attribut
    - → données publiques
    - → données confidentielles
    - → données secrètes

Exemples

Réglementation obligatoire (© Alban Gabillon / UPF)

- Décomposition en 3 tables mono-niveau
  - tout ce qui est classifié `Public` se retrouve dans la table publique, la table confidentielle et la table secrète
    → table `UNCLASSIFIED_AIRCRAFT`
  - tout ce qui est classifié `Confidentiel` se retrouve dans la table confidentielle et la table secrète
    → table `CONFIDENTIAL_AIRCRAFT`
  - tout ce qui est classifié `Secret` ne se trouve que dans la table secrète
    → table `SECRET_AIRCRAFT`

Exemples

Réglementation obligatoire (© Alban Gabillon / UPF)

- Table publique `UNCLASSIFIED_AIRCRAFT`

Name	Type	Speed	Range
F16	Supersonic	Mach 2.05	2000 km
Mirage	Supersonic	Mach 2.2	2000 km
Rafale	Supersonic	Mach 1.8	

Exemples

Réglementation obligatoire (© Alban Gabillon / UPF)

- Table publique `UNCLASSIFIED_AIRCRAFT`

Name	Type	Speed	Range
F16	Supersonic	Mach 2.05	2000 km
Mirage	Supersonic	Mach 2.2	2000 km
Rafale	Supersonic	Mach 1.8	`RESTRICTED`

→ `RESTRICTED` signifie que le niveau d'habilitation est insuffisant pour connaître la valeur de l'attribut

Exemples

Réglementation obligatoire (© Alban Gabillon / UPF)

- Table secrète CONFIDENTIAL_AIRCRAFT

Name	Type	Speed	Range
F16	Supersonic	Mach 2.05	2000 km
Mirage	Supersonic	Mach 2.2	2000 km
Rafale	Supersonic	Mach 1.8	4000 km
Firefox			6000 km

Exemples

Réglementation obligatoire (© Alban Gabillon / UPF)

- Table secrète CONFIDENTIAL_AIRCRAFT

Name	Type	Speed	Range
F16	Supersonic	Mach 2.05	2000 km
Mirage	Supersonic	Mach 2.2	2000 km
Rafale	Supersonic	Mach 1.8	4000 km
Firefox	Supersonic	Mach 2.5	6000 km

→ les valeurs de Type et de Speed pour le Firefox sont des mensonges (ou leurres) destinés à cacher l'existence de valeurs secrètes

Exemples

Réglementation obligatoire (© Alban Gabillon / UPF)

- Table secrète SECRET_AIRCRAFT

Name	Type	Speed	Range	Nuclear_bomb
F16	Supersonic	Mach 2.05	2000 km	NO
Mirage	Supersonic	Mach 2.2	2000 km	NO
Rafale	Supersonic	Mach 1.8	4000 km	YES
Firefox	Hypersonic	Mach 6	6000 km	YES

→ cette table contient toutes les données réelles

Exemples

Réglementation obligatoire (© Alban Gabillon / UPF)

- Implémentation du règlement multi-niveaux
  - → No Read-up
  - → No Write-down

Idée : Créer un rôle pour chaque niveau d'habilitation

```
CREATE ROLE PUBLIC;
CREATE ROLE CONFIDENTIEL;
CREATE ROLE SECRET;
```

Exemples

Réglementation obligatoire (© Alban Gabillon / UPF)

- Assignation des privilèges
  - rôle PUBLIC

    ```
    GRANT ALL ON UNCLASSIFIED_AIRCRAFT TO PUBLIC;
    ```

  - rôle CONFIDENTIEL

    ```
    GRANT SELECT ON UNCLASSIFIED_AIRCRAFT TO CONFIDENTIEL;
    GRANT ALL ON CONFIDENTIAL_AIRCRAFT TO CONFIDENTIEL;
    ```

  - rôle SECRET

    ```
    GRANT SELECT ON UNCLASSIFIED_AIRCRAFT TO SECRET;
    GRANT SELECT ON CONFIDENTIAL_AIRCRAFT TO SECRET;
    GRANT ALL ON SECRET_AIRCRAFT TO SECRET;
    ```

- Mécanismes de réplication d'un niveau bas vers un niveau haut assurés par des *triggers*

Plan du cours

Intégrité

- Quand on parle de sécurité informatique, le 1er objectif de sécurité qui nous vient à l'esprit est la **confidentialité**
  → prévention de la divulgation non autorisée de l'information

- L'**intégrité** est toutefois un objectif de sécurité tout aussi important, surtout dans le cadre d'un SGBD
  → *"l'intégrité de l'information s'attache à protéger l'information contre toute altération qui conduirait à mettre de l'information erronée à disposition des personnes autorisées"*

Intégrité & SGBD

- Contraintes d'Intégrité Référentielle (**CIR**)
  - générées "automatiquement" lors du passage du schéma E/A au schéma relationnel
  - correspond à la notion de `foreign key` en SQL

  ⇒ garantit qu'une référence "pointe" vers une information qui existe

  ex : `CONSTRAINT etudOk FOREIGN KEY (numEtu) REFERENCES Etudiant(numEtu)`

- Contraintes de **validité**

  ⇒ conditionne la création/modification d'un tuple à la validité de certaines propriétés

  ex : `CONSTRAINT noteOk CHECK ((note>=0) and (note<=20))`

- Concurrence d'accès ⇒ **transactions**

Systèmes transactionnels

- Une **transaction** telle qu'une réservation, un achat ou un paiement est mise en œuvre via :
  - → une suite d'opérations qui font passer la base de données d'un état A (antérieur à la transaction) à un état B (postérieur)
  - → des mécanismes permettent d'obtenir que cette suite soit à la fois ***A**tomique*, ***C**ohérente*, ***I**solée* et ***D**urable* (**ACID**)
- Le concept de transaction s'appuie sur la notion de point de synchronisation (*sync point*) qui représente un état stable du système informatique considéré (en particulier de ses données)

Systèmes transactionnels

- Propriétés ACID

atomicité la suite d'opérations est indivisible

en cas d'échec en cours d'une des opérations, la suite d'opérations doit être complètement annulée (`rollback`) quel que soit le nombre d'opérations déjà réussies

cohérence le contenu de la base de données à la fin de la transaction doit être cohérent sans pour autant que chaque opération durant la transaction donne un contenu cohérent

un contenu final incohérent doit entraîner l'échec et l'annulation de toutes opérations de la transaction

Systèmes transactionnels

- Propriétés ACID

isolation lorsque deux transactions A et B sont exécutées en même temps, les modifications effectuées par A ne sont ni visibles par B, ni modifiables par B tant que la transaction A n'est pas terminée et validée (`commit`)

durabilité une fois validé, l'état de la base de données doit être permanent, et aucun incident technique (ex : crash) ne doit pouvoir engendrer une annulation des opérations effectuées durant la transaction

Systèmes transactionnels

Lost updates

- Le système informatique d'une salle de cinéma (où les places ne sont pas numérotées) stocke le nombre de billets déjà vendus pour la séance
  - 100 billets ont déjà été vendus
  - la caisse 2 est en train d'en vendre trois autres
  - au même moment, la caisse 1 enregistre le remboursement de cinq billets, qui doivent donc être soustraits du total

Systèmes transactionnels

Lost updates

- Chaque transaction (caisse 1 & caisse 2) est composée de plusieurs actions élémentaires

caisse 1

1. lire le nombre de billets vendus (100)
2. rembourser 5 billets
3. calculer la nouvelle valeur (100-5=95)
4. écrire la nouvelle valeur (95)

caisse 2

1. lire le nombre de billets vendus (100)
2. vendre 3 billets
3. calculer la nouvelle valeur (100+3=103)
4. écrire la nouvelle valeur (103)

⇒ L'entrelacement de ces 2 transactions peut entraîner des erreurs

Systèmes transactionnels

Lost updates

- Exemple d'exécution ne respectant pas les propriétés ACID

caisse 1	nb billets vendus	caisse 2
	100	

Systèmes transactionnels

Lost updates

- Exemple d'exécution ne respectant pas les propriétés ACID

caisse 1	nb billets vendus	caisse 2
	100	
lire le nombre de billets vendus → résultat=100	100	

Systèmes transactionnels

Lost updates

- Exemple d'exécution ne respectant pas les propriétés ACID

caisse 1	nb billets vendus	caisse 2
	100	
lire le nombre de billets vendus → résultat=100	100	
	100	lire le nombre de billets vendus → résultat=100

Systèmes transactionnels

Lost updates

- Exemple d'exécution ne respectant pas les propriétés ACID

caisse 1	nb billets vendus	caisse 2
	100	
lire le nombre de billets vendus → résultat=100	100	
	100	lire le nombre de billets vendus → résultat=100
rembourser 5 billets calculer la nouvelle valeur : 100-5=95 écrire la nouvelle valeur (95)	95	

Systèmes transactionnels

Lost updates

- Exemple d'exécution ne respectant pas les propriétés ACID

caisse 1	nb billets vendus	caisse 2
	100	
lire le nombre de billets vendus → résultat=100	100	
	100	lire le nombre de billets vendus → résultat=100
rembourser 5 billets calculer la nouvelle valeur : 100-5=95 écrire la nouvelle valeur (95)	95	
	103	vendre 3 billets calculer la nouvelle valeur : 100+3=103 écrire la nouvelle valeur (103)

- Entrelacement ⇒ la 1ère maj est perdue ⇒ le résultat final est faux (103 au lieu de 98 billets effectivement vendus)

Systèmes transactionnels

Lost updates

- Une solution possible pour éviter les mises à jour perdues
  → utiliser un mécanisme de verrouillage des données

- Plusieurs types de verrous :

partagé un verrou partagé peut être détenu simultanément par un nombre arbitraire de transactions

exclusif un verrou exclusif ne peut être détenu que par une seule transaction à la fois ; il est également exclusif par rapport aux verrous partagés

- Une transaction qui requiert un verrou déjà utilisé est bloquée jusqu'à ce que le verrou soit relâché

Systèmes transactionnels

Lost updates

- Exécution en utilisant un verrou exclusif

caisse 1	nb billets vendus	caisse 2
	100	

Systèmes transactionnels

Lost updates

- Exécution en utilisant un verrou exclusif

caisse 1	nb billets vendus	caisse 2
	100	
demander un verrou exclusif	• 100	

Systèmes transactionnels

Lost updates

- Exécution en utilisant un verrou exclusif

caisse 1	nb billets vendus	caisse 2
	100	
demander un verrou exclusif	• 100	
	• 100	demander un verrou exclusif → bloquée

Systèmes transactionnels

Lost updates

- Exécution en utilisant un verrou exclusif

caisse 1	nb billets vendus	caisse 2
	100	
demander un verrou exclusif	• 100	
	• 100	demander un verrou exclusif → bloquée
lire le nombre de billets vendus → résultat=100	• 100	*waiting*...

Systèmes transactionnels

Lost updates

- Exécution en utilisant un verrou exclusif

caisse 1	nb billets vendus	caisse 2
	100	
demander un verrou exclusif	• 100	
	• 100	demander un verrou exclusif → bloquée
lire le nombre de billets vendus → résultat=100	• 100	*waiting*...
rembourser 5 billets calculer la nouvelle valeur : 100-5=95 écrire la nouvelle valeur (95)	• 95	*waiting*...

Systèmes transactionnels

Lost updates

- Exécution en utilisant un verrou exclusif

caisse 1	nb billets vendus	caisse 2
	100	
demander un verrou exclusif	• 100	
	• 100	demander un verrou exclusif → bloquée
lire le nombre de billets vendus → résultat=100	• 100	*waiting*...
rembourser 5 billets calculer la nouvelle valeur : 100-5=95 écrire la nouvelle valeur (95)	• 95	*waiting*...
relâcher le verrou	95 •	→ debloquée (obtient le verrou)

Systèmes transactionnels

Lost updates

- Exécution en utilisant un verrou exclusif

caisse 1	nb billets vendus	caisse 2
	100	
demander un verrou exclusif	• 100	
	• 100	demander un verrou exclusif → bloquée
lire le nombre de billets vendus → résultat=100	• 100	*waiting*...
rembourser 5 billets calculer la nouvelle valeur : 100-5=95 écrire la nouvelle valeur (95)	• 95	*waiting*...
relâcher le verrou	95 •	→ debloquée (obtient le verrou)
	95 •	lire le nombre de billets vendus → résultat=95

Systèmes transactionnels

Lost updates

- Exécution en utilisant un verrou exclusif

caisse 1	nb billets vendus	caisse 2
	100	
demander un verrou exclusif	• 100	
	• 100	demander un verrou exclusif → bloquée
lire le nombre de billets vendus → résultat=100	• 100	*waiting*...
rembourser 5 billets calculer la nouvelle valeur : 100-5=95 écrire la nouvelle valeur (95)	• 95	*waiting*...
relâcher le verrou	95 •	→ debloquée (obtient le verrou)
	95 •	lire le nombre de billets vendus → résultat=95
	98 •	vendre 3 billets calculer la nouvelle valeur : 95+3=98 écrire la nouvelle valeur (98)

Systèmes transactionnels

Lost updates

- Exécution en utilisant un verrou exclusif

caisse 1	nb billets vendus	caisse 2
	100	
demander un verrou exclusif	• 100	
	• 100	demander un verrou exclusif → bloquée
lire le nombre de billets vendus → résultat=100	• 100	*waiting*...
rembourser 5 billets calculer la nouvelle valeur : 100-5=95 écrire la nouvelle valeur (95)	• 95	*waiting*...
relâcher le verrou	95 •	→ debloquée (obtient le verrou)
	95 •	lire le nombre de billets vendus → résultat=95
	98 •	vendre 3 billets calculer la nouvelle valeur : 95+3=98 écrire la nouvelle valeur (98)
	98	relâcher le verrou

Systèmes transactionnels

Verrouillage de données

- L'utilisation de verrous permet de sérialiser les transactions
  - → l'exécution "entrelacée" des transactions est équivalente à leur exécution en série
  - → elles sont donc de fait isolées les unes des autres

Nb : De tels mécanismes de verrouillage sont fournis par tous les systèmes de données usuels

- pour les fichiers, par les verrous du système d'exploitation
- pour la mémoire partagée, par les sémaphores
- pour les bases de données, par des commandes spécifiques telles que la commande `SQL LOCK TABLE`

Systèmes transactionnels

Gestion des transactions

- En SQL Oracle pur
  - → le démarrage de la transaction est implicite (dès qu'on fait une requête `delete`, `insert` ou `update`)
  - → la transaction se termine par `COMMIT` ou `ROLLBACK`
- Si les appels SQL sont encapsulés dans un langage de haut niveau, il peut y avoir des particularités liées au langage et/ou à l'API utilisés

  ex : pour démarrer une transaction : `"BEGIN"`, `"START TRANSACTION"`, `"SET TRANSACTION"`, ...

Systèmes transactionnels

Isolation

- Certains SGBD proposent différents niveaux d'isolation
  - ***serializable*** → le plus fort : le système donne l'illusion que toutes les transactions sont exécutées en série
  - ***repeatable reads*** → autorise les lectures en parallèle (commandes `SELECT`), et rejette les écritures (en provoquant un `rollback` de la transaction) si une situation de mise à jour perdue est détectée

  ⇒ pb : *phantom reads*
  - ***read committed*** → les verrous de lecture sont relâchés dès que la commande `SELECT` est terminée

  ⇒ pb : *non-repeatable reads*
  - ***read uncommitted*** → niveau le plus bas : une transaction peut voir les modifications *not yet committed* d'autres transactions

  ⇒ pb : *dirty reads*

Systèmes transactionnels

Phantom reads

- Un *phantom read* apparaît quand, au sein d'une même transaction, 2 requêtes identiques sont exécutées et que la collection de lignes retournée par la 2nde est différente de celle de la 1ère

transaction 1	transaction 2
/* Query 1 */ SELECT * FROM users WHERE age BETWEEN 10 AND 30;	

Systèmes transactionnels

Phantom reads

- Un *phantom read* apparaît quand, au sein d'une même transaction, 2 requêtes identiques sont exécutées et que la collection de lignes retournée par la 2nde est différente de celle de la 1ère

transaction 1	transaction 2
/* Query 1 */ SELECT * FROM users WHERE age BETWEEN 10 AND 30;	
	/* Query 2 */ INSERT INTO users VALUES (3,'Bob',27); COMMIT;

Systèmes transactionnels

Phantom reads

- Un *phantom read* apparaît quand, au sein d'une même transaction, 2 requêtes identiques sont exécutées et que la collection de lignes retournée par la 2nde est différente de celle de la 1ère

transaction 1	transaction 2
/* Query 1 */ SELECT * FROM users WHERE age BETWEEN 10 AND 30;	
	/* Query 2 */ INSERT INTO users VALUES (3,'Bob',27); COMMIT;
/* Query 1 */ SELECT * FROM users WHERE age BETWEEN 10 AND 30;	

⇒ Le 2ème SELECT retourne 1 ligne supplémentaire

Systèmes transactionnels

Phantom reads

- *Note that Transaction 1 executed the same query twice. If the highest level of isolation were maintained, the same set of rows should be returned both times, and indeed that is what is mandated to occur in a database operating at the SQL SERIALIZABLE isolation level. However, at the lesser isolation levels, a different set of rows may be returned the second time.*

- *In the SERIALIZABLE isolation mode, Query 1 would result in all records with age in the range 10 to 30 being locked, thus Query 2 would block until the first transaction was committed. In REPEATABLE READ mode, the range would not be locked, allowing the record to be inserted and the second execution of Query 1 to include the new row in its results.*

Systèmes transactionnels

Non-repeatable reads

- Un *non-repeatable read* apparaît quand, au sein d'une même transaction, une même ligne est lue 2 fois mais est retournée avec des valeurs différentes

transaction 1	transaction 2
`/* Query 1 */` `SELECT * FROM users WHERE id = 1;`	

Systèmes transactionnels

Non-repeatable reads

- Un *non-repeatable read* apparaît quand, au sein d'une même transaction, une même ligne est lue 2 fois mais est retournée avec des valeurs différentes

transaction 1	transaction 2
`/* Query 1 */` `SELECT * FROM users WHERE id = 1;`	

	`/* Query 2 */`
	`UPDATE users SET age = 21 WHERE id = 1;`
	`COMMIT;`
	`/* in multiversion concurrency control,`
	`or lock-based READ COMMITTED */`

Systèmes transactionnels

Non-repeatable reads

- Un *non-repeatable read* apparaît quand, au sein d'une même transaction, une même ligne est lue 2 fois mais est retournée avec des valeurs différentes

transaction 1	transaction 2
`/* Query 1 */` `SELECT * FROM users WHERE id = 1;`	

	`/* Query 2 */`
	`UPDATE users SET age = 21 WHERE id = 1;`
	`COMMIT;`
	`/* in multiversion concurrency control,`
	`or lock-based READ COMMITTED */`

`/* Query 1 */`
`SELECT * FROM users WHERE id = 1;`
`COMMIT;`
`/* lock-based REPEATABLE READ */`	

⇒ Le 2ème SELECT retourne des valeurs différentes pour cette ligne

Systèmes transactionnels

Non-repeatable reads

- *In this example, Transaction 2 commits successfully, which means that its changes to the row with id 1 should become visible. However, Transaction 1 has already seen a different value for age in that row. At the SERIALIZABLE and REPEATABLE READ isolation levels, the DBMS must return the old value. At READ COMMITTED and READ UNCOMMITTED, the DBMS may return the updated value ; this is a non-repeatable read.*
- *There are two basic strategies used to prevent non-repeatable reads. The first is to delay the execution of Transaction 2 until Transaction 1 has committed or rolled back. This method is used when locking is used, and produces the serial schedule T1, T2. A serial schedule does not exhibit non-repeatable reads behaviour.*
- *In the other strategy, as used in multiversion concurrency control, Transaction 2 is permitted to commit first, which provides for better concurrency. However, Transaction 1, which commenced prior to Transaction 2, must continue to operate on a past version of the database — a snapshot of the moment it was started. When Transaction 1 eventually tries to commit, the DBMS checks if the result of committing Transaction 1 would be equivalent to the schedule T1, T2. If it is, then Transaction 1 can proceed. If it cannot be seen to be equivalent, however, Transaction 1 must roll back with a serialization failure.*

Systèmes transactionnels

Non-repeatable reads

- *Using a lock-based concurrency control method, at the REPEATABLE READ isolation mode, the row with ID = 1 would be locked, thus blocking Query 2 until the first transaction was committed or rolled back. In READ COMMITTED mode, the second time Query 1 was executed, the age would have changed.*
- *Under multiversion concurrency control, at the SERIALIZABLE isolation level, both SELECT queries see a snapshot of the database taken at the start of Transaction 1. Therefore, they return the same data. However, if Transaction 1 then attempted to UPDATE that row as well, a serialization failure would occur and Transaction 1 would be forced to roll back.*
- *At the READ COMMITTED isolation level, each query sees a snapshot of the database taken at the start of each query. Therefore, they each see different data for the updated row. No serialization failure is possible in this mode (because no promise of serializability is made), and Transaction 1 will not have to be retried.*

Systèmes transactionnels

Dirty reads

- Un *dirty read* apparaît quand une transaction peut voir une ligne modifiée par une autre transaction encore en cours d'exécution (*not yet committed*)

transaction 1	transaction 2
`/* Query 1 */` `SELECT * FROM users WHERE id = 1;`	

Systèmes transactionnels

Dirty reads

- Un *dirty read* apparaît quand une transaction peut voir une ligne modifiée par une autre transaction encore en cours d'exécution (*not yet committed*)

transaction 1	transaction 2
`/* Query 1 */` `SELECT * FROM users WHERE id = 1;`	

	`/* Query 2 */`
	`UPDATE users SET age = 21 WHERE id = 1;`
	`/* No commit here */`

Systèmes transactionnels

Dirty reads

- Un *dirty read* apparaît quand une transaction peut voir une ligne modifiée par une autre transaction encore en cours d'exécution (*not yet committed*)

transaction 1	transaction 2
`/* Query 1 */` `SELECT * FROM users WHERE id = 1;`	

	`/* Query 2 */`
	`UPDATE users SET age = 21 WHERE id = 1;`
	`/* No commit here */`

`/* Query 1 */` `SELECT * FROM users WHERE id = 1;`	

Systèmes transactionnels

Dirty reads

- Un *dirty read* apparaît quand une transaction peut voir une ligne modifiée par une autre transaction encore en cours d'exécution (*not yet committed*)

transaction 1	transaction 2
`/* Query 1 */` `SELECT * FROM users WHERE id = 1;`	

	`/* Query 2 */`
	`UPDATE users SET age = 21 WHERE id = 1;`
	`/* No commit here */`

`/* Query 1 */` `SELECT * FROM users WHERE id = 1;`	
	`ROLLBACK; /* lock-based DIRTY READ */`

⇒ Donnée erronée dans la transaction 1 car elle a lu une valeur *uncommitted* et la transaction 2 a annulé la modification de cette donnée

Conclusion

- Isolation Levels vs Read Phenomena

Isolation level	Dirty reads	Non-repeatable reads	Phantoms
Read Uncommitted	may occur	may occur	may occur
Read Committed	-	may occur	may occur
Repeatable Read	-	-	may occur
Serializable	-	-	-

- Isolation Levels vs Locks

Isolation level	Write Lock	Read Lock	Range Lock
Read Uncommitted	-	-	-
Read Committed	V	(1)	-
Repeatable Read	V	V	-
Serializable	V	V	V

- "V" indicates that the method locks for that operation, keeping that lock till the end of the transaction containing that operation

(1) read (shared) locks are released immediately after the SELECT operation is performed

Conclusion

- Ces différents problèmes d'intégrité (cf. *lost updates*, *phantom reads*, *non-repeatable reads*, *dirty reads*) sont incontournables dès lors que plusieurs composants logiciels accèdent de manière concurrente à des ressources communes !
  - → transactions dans un SGBD
  - → applications accédant à des fichiers partagés
  - → processus utilisant une mémoire partagée
  - → services faisant appel à des ressources réseau communes
  - → ...

⇒ Besoin de mécanismes de synchronisation

nb : il peut être nécessaire de gérer les "`rollbacks`" au niveau applicatif pour réitérer la transaction

Plan du cours

La sécurité en informatique

- **Authentifier** les utilisateurs ou les programmes avec des signatures
- Contrôler les **autorisations** d'accès aux ressources (système & utilisateur) d'une entité authentifiée
  - *Ex : empêcher des lectures, suppressions ou modifications non autorisées sur des fichiers*
- Assurer par le **cryptage** la confidentialité des transmissions ou des données stockées

Le soucis de la sécurité

- Dès sa conception, Java a attaché beaucoup d'importance à la sécurité
- Les classes Java peuvent être chargées dynamiquement depuis des machines non sûres et il faut un moyen de contrôler les actions exécutées par ces classes
- En particulier, l'exécution des applets récupérées sur le Web présente un grand risque

Interdire pour protéger

- Pour effectuer la protection des ressources, Java interdit aux classes Java non sûres (essentiellement celles qui viennent d'une autre machine) d'effectuer des opérations potentiellement dangereuses :
  - obtenir des informations sur l'utilisateur ou la machine locale
  - lire/écrire sur le disque local
  - se connecter sur une tierce machine non connue de l'utilisateur
  - ...

Le bac à sable des applets

- Par défaut les applets s'exécutent dans un environnement spécifique, le "bac à sable" (*sandbox*), duquel elles ne peuvent sortir et qui leur interdit toute action dangereuse
- Mais les contraintes imposées par le bac à sable sont trop strictes pour certaines applets qui souhaitent obtenir des informations locales ou écrire des données sur les disques locaux
- Depuis Java 2 (aka Java 1.2) cette politique de sécurité a été assouplie

Objectif du cours

- Les exemples suivants montrent les restrictions imposées par le bac à sable
- Il faudra "contourner" ces restrictions si on veut faire fonctionner les applications
- Mais il ne faut pas pour cela ouvrir la porte aux utilisateurs mal intentionnés

Sécurité Java : gestion des permissions

Comment permettre **juste ce qu'il faut** pour qu'une application ou applet (et pas les autres) puisse effectuer des opérations qui lui sont interdites normalement ?

Limites du bac à sable : 1er exemple

- Application qui pose des QCM et gère les résultats
- Pour cela, l'utilisateur se connecte à une adresse Web et récupère une page qui contient une applet qui affiche les questions
- Pour obtenir l'identité de la personne qui répond, l'applet contient l'instruction suivante :
  `System.getProperty("user.name");`
- Le bac à sable dans lequel s'exécutent les applets interdit la lecture de la propriété système `"user.name"`

Limites du bac à sable : 2ème exemple

- Cette même application range les résultats des étudiants dans une base de données
- Le SGBD est placé sur une autre machine que le serveur Web
- L'applet se voit refuser l'accès à ce SGBD car le bac à sable interdit les connexions réseaux sur une autre machine que celle du serveur Web

Limites du bac à sable : 3ème exemple

- Une applet souhaite écrire sur le disque de l'utilisateur un fichier qui contient des données qui seront utiles à l'utilisateur la prochaine fois qu'il se connectera
- La politique de sécurité liée au bac à sable refuse à l'applet l'écriture d'un fichier sur le disque local de l'utilisateur
- En fait une applet ne peut que lire des fichiers qui sont placés dans le même répertoire ou dessous (et placés dans le même jar si elle est dans un jar)

Il n'y pas que les applets...

- Les restrictions imposées par le bac à sable des applets se retrouvent aussi pour les applications qui chargent des classes dynamiquement depuis une autre machine (RMI et Java Web Start en particulier)
- Idem pour les applications qui décident, pour une raison quelconque, d'activer un gestionnaire de sécurité

Plan du cours

La plate-forme Java

- 3 composants
  - la machine virtuelle Java (JVM pour *Java Virtual Machine*)
  - le langage Java
  - l'ensemble des API Java (les classes systèmes)
- Programme Java = ensemble de classes fondées sur les classes systèmes
- Compilation = traduction du code Java en bytecode ("assembleur" portable de haut niveau)
- Exécution = interprétation du bytecode par la JVM

Environnement d'exécution

- Java remplace la notion d'exécutable par celle de composant logiciel
- Un programme Java ⇒ formes différentes selon le but visé :
  - **Application Java** = version Java d'un exécutable
    point d'entrée = main(), démarré dans une JVM dédiée,...
  - **Applet Java** = composant logiciel qui s'intègre à un navigateur web
    applet chargée dynamiquement par une JVM
  - **Servlet Java** = composant logiciel qui s'intègre à un serveur web
  - Autres : Enterprise Java Beans, composant OSGi,...

Environnement d'exécution

- Chaque "forme" est associée à un environnement d'exécution particulier
  - qui fournit des services
  - qui impose des restrictions particulières
- Points à retenir :
  - La JVM peut être considérée comme un véritable ordinateur muni d'un système d'exploitation
  - Un programme Java est dynamiquement chargé dans la JVM pour être exécuté
  - Il est exécuté en concurrence avec les autres programmes déjà en cours d'exécution dans la même JVM
- Le **mécanisme de chargement des classes** (le *classloader*) est central

Java & sécurité

- Dans le langage Java et la JVM
- Les API standards
- Les API et les outils dédiés à la sécurité

La sécurité dans le langage

- Vérifications à la **compilation** :
  - Java est fortement typé
  - pas d'arithmétique des pointeurs
  - les variables non initialisées sont inutilisables
  - `return` obligatoire à la fin des méthodes non `void`
  - protection des variables d'état (`private`,...)
  - possibilité de déclarer `final` les variables, méthodes et classes

- Vérifications à l'**exécution** :
  - contrôle des débordements dans les tableaux
  - contrôle des *casts*
  - vérification des classes au chargement

La sécurité dans les API

- Les API liées à la sécurité permettent de
  - **délimiter ce qui est autorisé** pour chaque programme Java, selon
    - le lieu d'où les classes ont été chargées
    - l'utilisateur qui les a signées
    - l'utilisateur qui l'exécute (étudié plus loin lors de l'étude de JAAS ; pas pris en compte pour cette partie)
  - **protéger la confidentialité** des informations par le cryptage
  - **authentifier des données** via des signatures

Écrire des programmes sûrs

- Favoriser l'encapsulation des données : les attributs doivent être `private`, sauf raison contraire
- Éviter autant que possible les membres `protected`
- Éviter de passer des références vers des variables sensibles en sortie ou en entrée des méthodes
  - ⇒ cloner (ou copier) les valeurs avant de les passer
- Déclarer `final` les méthodes ou les classes dont le fonctionnement ne doit pas être modifié

Vérification du *bytecode*

- Il fait partie de la JVM
- Son rôle est d'examiner le *bytecode* des classes au moment de leur transformation en objet `Class` par le chargeur de classe
- Il vérifie que la structure des classes chargées par la JVM est correcte
- En plus des avantages pour la sécurité, il évite ainsi à la JVM d'effectuer certaines vérification à l'exécution (par exemple, les dépassements de capacité de la pile), ce qui améliore les performances

Vérification du *bytecode*

- Vérification en 4 passes :
  1. Vérification de la structure du fichier
  2. Vérifications qui ne dépendent pas du code particulier des méthodes
  3. Vérification du code de chaque méthode
  4. Vérifications sur le code des méthodes, qui sont repoussées pour des raisons d'efficacité jusqu'au moment où le code est exécuté pour la première fois

Vérification du *bytecode*

- Passes 1 & 2 :
  - Structure du fichier
    - nombre "magique" correct au début (`0xCAFEBABE`)
    - structure correcte du pool des constantes
  - Vérifications qui ne dépendent pas du code particulier des méthodes
    - pas de classe fille d'une classe `final`
    - pas de méthode `final` redéfinie
    - toute classe a bien une classe mère (sauf `Object`)
    - les références vers le pool des constantes sont valables

Vérification du *bytecode*

- Passe 3 :
  - Jusqu'à maintenant les vérifications ne prenaient en compte que la classe vérifiée ; les passes 3 et 4 prennent en compte les **classes utilisées** par la classe vérifiée ⇒ nécessite une analyse du cheminement d'exécution
    - *NB : La répartition des tâches entre les passes 3 et 4 n'est pas complètement spécifiée et peut dépendre des JVM*
  - Le principe général est qu'on essaie de repousser dans la passe 4 les vérifications qui nécessiteraient le chargement de nouvelles classes
  - Par exemple, si le profil de `m` est `"B m()"`, alors `"B b=m();"` peut être vérifié par la passe 3 sans charger la classe `B` ; mais `"A b=m();"` va nécessiter le chargement des classes `A` et `B` pour s'assurer que `B` est une classe fille de `A`

Vérification du *bytecode*

- Vérifications effectuées à la passe 3 :
  - les variables locales ne sont pas utilisées sans avoir été initialisées
  - les méthodes sont appelées avec des paramètres corrects (nombre et type) et retournent les bons types
  - les accès aux membres (`public`, `protected`,...) sont autorisées
  - les affectations sont effectuées avec les bons types
- Passe 4 :
  - Les vérifications effectuées sont celles de la passe 3 qui nécessitent le chargement de classes
  - La JVM peut alors remplacer les instructions qui ont déclenché la vérification par d'autres instructions spéciales de telle sorte qu'aucune vérification ne sera plus déclenchée par la suite

Vérification du *bytecode*

- Classes vérifiées par le vérificateur de *bytecode* :
  - Les classes de l'API standard (et des extensions) ne sont pas vérifiées
  - Depuis la version 1.2, les classes venant du *classpath* sont vérifiées

Protection par le *classloader*

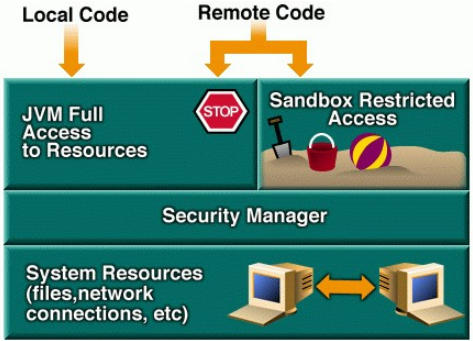
- Les chargeurs de classes sont les piliers de la gestion de la sécurité en Java 2
- Lors du chargement des classes ils mettent en place tout le contexte qui sera utilisé par les gestionnaires de sécurité pour délimiter ce qui sera autorisé pendant l'exécution
- En particulier, ils associent les classes chargées à un domaine de protection (voir suite du cours)
- Les chargeurs de classes ne seront pas étudiés en détails ici (voir un autre cours)

Espaces de noms liés aux *classloaders*

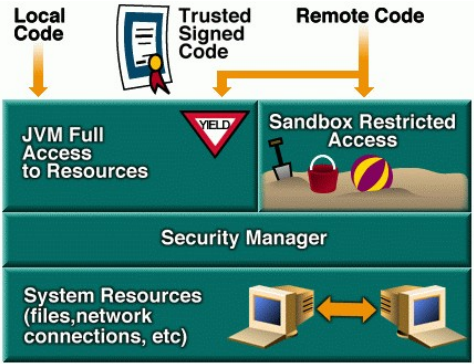
- Un chargeur de classes définit un **espace de nom** : 2 classes de même nom chargées par 2 chargeurs de classes différents sont considérées comme différentes
  - les navigateurs créent un chargeur de classes différent pour chaque *codeBase*
  - ainsi, par exemple, 2 applets venant de 2 URL différentes ont des chargeurs de classes différents et ne partagent pas les mêmes classes distantes (même si elles ont le même nom)
- La politique de sécurité de Java a évolué au cours des différentes versions : JDK 1.0, JDK 1.1 et Java 1.2
  - chaque nouvelle version a ajouté de la souplesse aux possibilités offertes aux développeurs et administrateurs

Les versions de la politique de sécurité de Java 1

JDK 1.0 ⇒ tout ou (presque) rien



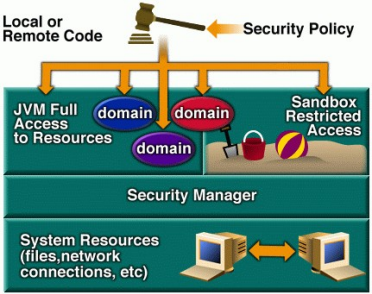
JDK 1.1 ⇒ tout (**aussi pour le code signé**) ou (presque) rien



(Images issues du tutoriel en ligne de Sun)

La politique de sécurité de Java 2

- Plusieurs domaines de sécurité avec plus ou moins d'autorisations (de tout jusqu'à presque rien)
- Les domaines de sécurité sont déterminés par les chargeurs de classe



Plan du cours

Introduction à la politique de sécurité de Java 2

- En Java, les droits d'un code ne sont pas écrits en dur dans le code
- Ils sont définis par une **politique de sécurité** enregistrée à part dans des fichiers
- Exemple (ne pas oublier les ";") :

```
keystore ".keystore";

grant signedBy "toto" {
  permission java.io.FilePermission "${user.home}${/}-","read";
};

grant codeBase "http://www.univ-pau.fr/~toto/*" {
  permission java.util.PropertyPermission "user.home","read";
};
```

Introduction à la politique de sécurité de Java 2

Domaine de sécurité

- **Quand elle est chargée dans la JVM**, une classe se voit associer un **domaine de sécurité**
- Un domaine de sécurité ajoute des permissions à une classe (en plus de ce que permet le bac à sable) ; c'est une sorte de bac à sable élargi
- Un domaine de sécurité est déterminé par :
  - l'origine de la classe : d'où vient-elle, qui l'a signée
  - la politique de sécurité **au moment où la classe est chargée** par le chargeur de classes
- La politique de sécurité peut être modifiée en cours d'exécution, mais ça ne change pas les permissions associées aux classes déjà chargées

Introduction à la politique de sécurité de Java 2

Exemple

- L'utilisateur qui a lancé l'exécution a ceci dans son fichier de police de sécurité :

```
grant signedBy "toto" {
  permission java.io.FilePermission "${user.home}${/}-","read";
};
```

- Alors, si une classe signée par `toto` est chargée en mémoire, le domaine de sécurité qui lui est associé contiendra la permission de lire les fichiers placés dans toute l'arborescence de son `home directory`

Introduction à la politique de sécurité de Java 2

Toutes les API sont concernées

- Toute action potentiellement dangereuse effectuée par une méthode des API Java est contrôlée par le gestionnaire de sécurité
- Les paquetages `java.io` et `java.net` sont particulièrement concernés, mais aussi les autres paquetages comme `java.lang`, `java.awt` ou `javax.swing`
- API et outils spécifiquement destinés à la sécurité
  - Paquetage `java.security` et ses sous paquetages : classes pour les permissions, les signatures, les polices de sécurité,...
  - Outils pour la sécurité fournis avec le SDK : `keytool`, `jarsigner`, `policytool`

Plan du cours

Les acteurs de la politique de sécurité

- 4 acteurs de la sécurité en Java :
  - → la politique de sécurité
  - → les domaines de sécurité
  - → le gestionnaire de sécurité et le contrôleur d'accès
  - → le contexte d'appel de méthodes
- La **politique de sécurité** détermine le **domaine de sécurité** d'une classe au moment de son chargement
- Si une méthode veut exécuter une action "dangereuse", alors le **gestionnaire de sécurité** charge le **contrôleur d'accès** de vérifier qu'elle en a le droit dans le **contexte de l'appel**

Les acteurs de la politique de sécurité

Politique de sécurité

- Par défaut elle est déterminée par la lecture d'un fichier de propriétés au démarrage de la JVM
- Elle est utilisée au moment du chargement des classes pour créer des **domaines de sécurité**
- La vérification des droits d'une classe n'utilise ensuite plus **que** ces domaines de sécurité
- La méthode `refresh()` permet de relire le fichier de police de sécurité (pour l'attribution des domaines des nouvelles classes chargées)
- NB : Une politique de sécurité est représentée par une instance de la classe `Policy`

Les acteurs de la politique de sécurité

Domaine de sécurité

- Une classe appartient à un et un seul domaine déterminé par la politique de sécurité au moment de son chargement et par son *code source*, c'est-à-dire l'origine du code
- Un domaine de protection est composé d'un codesource, d'une collection de permissions, d'un classloader et d'un tableau de *principals* (servent à identifier un utilisateur)
- Représenté par la classe `ProtectionDomain`

Les acteurs de la politique de sécurité

Notion de *CodeSource*

- En Java 2, **l'origine du code** d'une classe détermine à lui seul son domaine de sécurité (voir aussi JAAS)
- Cette origine est constituée
  - de l'URL d'où a été chargée la classe
  - du signataire de la classe, avec les certificats qui ont été utilisés pour vérifier la signature
- Représenté par la classe `CodeSource`

Les acteurs de la politique de sécurité

Domaine système

- Le domaine "système" est formé des classes chargées par le chargeur de classe primordial (celui qui charge les classes au démarrage de Java)
- Les classes du domaine système ne sont pas contrôlées
- Actuellement, toutes les classes du JDK sont placées dans le domaine système
- Les applets ou les applications appartiennent à d'autres domaines *code source*

Les acteurs de la politique de sécurité

Gestionnaire de sécurité

- Il contrôle l'accès aux ressources "protégées"
- Représenté par la classe `java.lang.SecurityManager` ou par une classe descendante
- En fait, le *security manager* délègue maintenant le travail au contrôleur d'accès (classe `AccessControler` depuis Java 2)
  - ⇒ Extrait du code de `SecurityManager`

```
public void checkWrite(String file) {
  checkPermission(new FilePermission(file,"write"));
}

public void checkPermission(Permission perm) {
  AccessController.checkPermission(perm);
}
```

Les acteurs de la politique de sécurité

Quel gestionnaire de sécurité ?

- Ordinairement, il est installé par le contexte d'exécution (par exemple, par le navigateur Internet ou par une option de la commande `java`)
- Par défaut, les applications locales (lancées par la commande `java`) n'installent pas de gestionnaire de sécurité
- Aucun contrôle d'accès n'est donc effectué pour les applications locales, même pour les classes distantes chargées par l'application, par exemple, en utilisant un `URLClassLoader`

Les acteurs de la politique de sécurité

Quel gestionnaire de sécurité ?

- On peut activer un gestionnaire de sécurité au **lancement d'une application** via une option de la commande `java` :
  - utiliser le gestionnaire par défaut (bac à sable, instance de `java.lang.SecurityManager`)

    `java` **-Djava.security.manager** ... <classe>

  - utiliser un gestionnaire particulier :

    `java` **-Djava.security.manager=MonSecurityMgr** ... <classe>

Les acteurs de la politique de sécurité

Quel gestionnaire de sécurité ?

- On peut également installer un gestionnaire de sécurité en cours d'exécution (pour des RMI sur cet exemple) :

```
System.setSecurityManager(new RMISecurityManager())
```

- Cet appel nécessite le droit

```
java.lang.RuntimePermission("setSecurityManager")
```

- Pour récupérer le gestionnaire de sécurité actuellement activé :

```
System.getSecurityManager()
```

Les acteurs de la politique de sécurité

Déboguer les problèmes liés aux permissions

- Pour les applications, la propriété `java.security.debug` offre plusieurs options pour faire afficher les problèmes liées aux permissions ; ces options s'affichent en donnant par exemple :

```
java -Djava.security.debug=help
```

- Pour les applets, il faut faire afficher la console Java du navigateur ou du plugin Java de Sun

Les acteurs de la politique de sécurité

Contrôleur d'accès

- C'est lui qui décide si un accès à une ressource système est autorisé
- Il tient compte :
  - du contexte d'appel de la méthode qui veut accéder à la ressource système
  - du domaine de sécurité de **toutes les classes** qui sont dans le contexte d'appel
- C'est une instance de la classe `AccessController` qui ne contient que des méthodes statiques (`checkPermission` en particulier)

Les acteurs de la politique de sécurité

Contexte d'appel

- Il est utilisé pour vérifier les droits d'une méthode
- Il est formé des classes dont les méthodes sont dans les piles d'exécution des différents *threads* qui ont conduit à l'appel de la méthode
- Pour qu'une méthode ait un droit, il faut que **tous les domaines** des classes du contexte d'appel aient ce droit
- Représenté par la classe `AccessControlContext`

Les acteurs de la politique de sécurité

Contexte d'appel

- Extrait du code de `FileOutputStream`

```
public FileOutputStream(String name,boolean append)
  throws FileNotFoundException
{
  SecurityManager security = System.getSecurityManager();
  if (security != null) {
    security.checkWrite(name);
  }
  fd = new FileDescriptor();
  if (append) {
    openAppend(name);
  }
  else {
    open(name);
  }
}
```

Les acteurs de la politique de sécurité

Contexte d'appel

- Extrait du code de `SecurityManager`

```
public void checkWrite(String file) {
  checkPermission(new FilePermission(file,"write"));
}

public void checkPermission(Permission perm) {
  AccessController.checkPermission(perm);
}
```

Les acteurs de la politique de sécurité

AccessControlException

- Si un appel est rejeté par le contrôleur d'accès, une exception `AccessControlException` (paquetage `java.security`) est levée
- Cette classe hérite de la classe `java.lang.SecurityException` qui elle-même hérite de `RuntimeException`
- Il n'est donc pas obligatoire de déclarer cette exception dans la définition des méthodes

Les acteurs de la politique de sécurité

Contrôle d'accès

- Pile d'exécution (méthodes) & contexte d'appel (classes)

Méthode	Classe	Domaine	Permissions
$m_A()$	$Classe_A$	dom_A	$p_{A_1}, p_{A_2}, \ldots$

Les acteurs de la politique de sécurité

Contrôle d'accès

- Pile d'exécution (méthodes) & contexte d'appel (classes)

Méthode	Classe	Domaine	Permissions
$m_A()$	$Classe_A$	dom_A	$p_{A_1}, p_{A_2}, \ldots$
$m_B()$	$Classe_B$	dom_B	$p_{B_1}, p_{B_2}, \ldots$

Les acteurs de la politique de sécurité

Contrôle d'accès

- Pile d'exécution (méthodes) & contexte d'appel (classes)

Méthode	Classe	Domaine	Permissions
$m_A()$	$Classe_A$	dom_A	$p_{A_1}, p_{A_2}, \ldots$
$m_B()$	$Classe_B$	dom_B	$p_{B_1}, p_{B_2}, \ldots$
$m_C()$	$Classe_C$	dom_C	$p_{C_1}, p_{C_2}, \ldots$

Les acteurs de la politique de sécurité

Contrôle d'accès

- Pile d'exécution (méthodes) & contexte d'appel (classes)

Méthode	Classe	Domaine	Permissions
$m_A()$	$Classe_A$	dom_A	$p_{A_1}, p_{A_2}, \ldots$
$m_B()$	$Classe_B$	dom_B	$p_{B_1}, p_{B_2}, \ldots$
$m_C()$	$Classe_C$	dom_C	$p_{C_1}, p_{C_2}, \ldots$
read()	FileReader	système	all

Les acteurs de la politique de sécurité

Contrôle d'accès

- Pile d'exécution (méthodes) & contexte d'appel (classes)

Méthode	Classe	Domaine	Permissions
$m_A()$	$Classe_A$	dom_A	$p_{A_1}, p_{A_2}, \ldots$
$m_B()$	$Classe_B$	dom_B	$p_{B_1}, p_{B_2}, \ldots$
$m_C()$	$Classe_C$	dom_C	$p_{C_1}, p_{C_2}, \ldots$
read()	FileReader	système	all
checkPermission()	AccessController	système	all

- Les domaines dom_A, dom_B et dom_C doivent tous contenir la permission `"read"`, sinon lancement d'une exception de type `AccessControlException`

Les acteurs de la politique de sécurité

Contexte d'appel : exemple 1/4

- On regroupe ces 2 classes dans le fichier `main.jar` et on déclare `Main` comme classe principale [17]

```
package main;
public class Main {
  public static void main(String[] args) {
    Truc.test();
  }
}

package main;
import java.io.File;
public class Truc {
  public static void test() {
    File f=new File("/tmp/my-test-file");
    f.delete();
  }
}
```

17. `> jar cvfe main.jar main.Main main/*.class`

Les acteurs de la politique de sécurité

Contexte d'appel : exemple 2/4

- À l'exécution, on finit par exécuter `f.delete()`
- Le contrôleur d'accès vérifie que l'effacement du fichier nommé `/tmp/my-test-file` est bien autorisé
  - La vérification est demandée par la classe `File` qui est une classe système
  - La pile d'appels contient à ce moment les méthodes `Main.main`, `Truc.test` et `File.delete`
  - Les classes systèmes ont bien sûr tous les droits ⇒ `File` a le droit d'effacer le fichier (au sens Java, le SE ne sera pas forcément d'accord...)
  - **Règle d'intersection des domaines** ⇒ les classes `Main` et `Truc` doivent également posséder le droit d'effacement

Les acteurs de la politique de sécurité

Contexte d'appel : exemple 3/4

- On installe un gestionnaire de sécurité :

```
java -Djava.security.manager -jar main.jar
```

- Mais on n'installe pas de politique de sécurité particulière ⇒ bac à sable par défaut
- Conséquence :
  - nos classes locales n'ont aucun droit particulier

  ⇒ le programme est interrompu par une exception de sécurité

Les acteurs de la politique de sécurité

Contexte d'appel : exemple 4/4

- On crée une politique de sécurité (fichier `policy.txt`) :

```
grant codeBase "file:main.jar" {
  permission java.io.FilePermission "/tmp/my-test-file", "delete";
};
```

- On charge cette politique de sécurité :

```
java -Djava.security.manager -Djava.security.policy=policy.txt -jar main.jar
```

- Cette fois, le programme fonctionne

  les deux classes sont dans `main.jar` (dossier courant)

  ⇒ elles appartiennent au domaine décrit par la politique de sécurité

  ⇒ elles ont le droit d'effacer le fichier

  ⇒ le contrôleur d'accès autorise donc l'effacement

Plan du cours

Les fichiers de la politique de sécurité

Fichiers de configuration par défaut

- Les emplacements des fichiers de politique de sécurité sont indiqués (sous Unix) dans le fichier de propriétés

  `java.home/lib/security/java.security`

  où `java.home` est le répertoire où a été installé le programme `jre` (*Java Runtime Environment*), en général le sous-répertoire `jre` du répertoire où a été installé le SDK

- Ces emplacements sont indiqués par les propriétés `policy.url.n` (n=1,2,...) du fichier de propriétés

- La convention est de donner l'extension `.policy` aux fichiers de politique de sécurité

Les fichiers de la politique de sécurité

Emplacements par défaut

- Politiques "système" et "utilisateur" :

```
policy.url.1=file:${java.home}/lib/security/java.policy
policy.url.2=file:${user.home}/.java.policy
```

  NB : `${java.home}` représente la valeur de la propriété `java.home`

- S'il n'existe pas de fichiers de politique, la politique de sécurité correspond aux restrictions du bac à sable des applets (presque rien n'est autorisé)

Les fichiers de la politique de sécurité

Ajouter une politique supplémentaire

- 2 moyens pour ajouter un fichier de permissions :
  - option `-Djava.security.policy=myPolicy` de la commande `java` (ou `-J-D...` pour `appletviewer`)
    - → "=" ajoute les règles à la politique par défaut
    - → "==" remplace la politique par défaut
  - ajouter une propriété `policy.url.n` dans un fichier de propriétés ; par exemple :

    `policy.url.3=`**file:${user.home}/myPolicy**
- La 1ère solution est la meilleure pendant les phases de test...

Les fichiers de la politique de sécurité

Contenu des fichiers de politique de sécurité

- Un fichier de politique de sécurité contient des entrées du type `grant` (`signedBy` et `codeBase` sont optionnels) :

```
grant signedBy "signataires", codeBase "url" {
  permission1;
  permission2;
};
```

NB : Ne pas oublier les ";"

- Il peut aussi contenir une entrée `keystore` :

```
keystore "emplacement", "type";
```

NB : type `jks` par défaut (format de SUN)

Les fichiers de la politique de sécurité

Exemple de politique de sécurité

```
// Un commentaire

keystore ".keystore";

grant signedBy "toto" {
  permission java.io.FilePermission "${user.home}${/}-", "read";
};

grant codeBase "http://www.univ-pau.fr/~toto/*" {
  permission java.util.PropertyPermission "user.home", "read";
};
```

Les fichiers de la politique de sécurité

Entrée grant

- Une permission commence par le mot clé `grant` suivi optionnellement d'un `signedBy` et/ou `codeBase` (dans un ordre quelconque)
- Exemples :

```
grant
grant signedBy "signataire[,signataire2,...]"
grant codeBase "uneURL"
grant signedBy "signataire",codeBase "uneURL"
```

  → si plusieurs signataires, le code doit être signé par **tous les signataires** (relation **et**, pas ou)

  → codeBase limite la permission au code qui vient de cette URL

Les fichiers de la politique de sécurité

Format des URL pour le codeBase

- Différentes significations selon la fin de l'URL :
  - "/" désigne toutes les classes (pas les JAR) situées dans le répertoire placé avant le "/"
  - "/*" désigne toutes les classes (y compris les classes dans les fichiers JAR) situées dans le répertoire
  - "/-" désigne toutes les classes (y compris les classes dans les fichiers JAR) situées dans l'**arborescence** du répertoire
- Si le codeBase désigne un répertoire, il s'agit de la "racine" du chemin et les classes se trouvent dessous avec le chemin correspondant à leur paquetage
- Dans les URL, le séparateur est "/" (quel que soit l'OS)

Les fichiers de la politique de sécurité

Entrée `permission`

- Le format standard est :

  `permission classePermission "but", "action";`

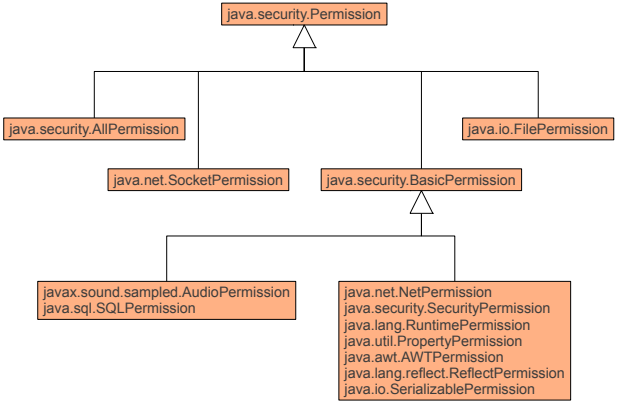
  - `classePermission` indique le type de permission que l'on donne (nom d'une classe de permission)
  - `but` indique sur quel objet on donne la permission
  - `action` indique quel type d'action on autorise sur le `but`

- Certaines permissions peuvent ne pas avoir d'action, ni même de but

- Une permission peut comporter une clause `signedBy "noms..."` (`classePermission` doit être signée)

Les fichiers de la politique de sécurité

Quelques classes de permissions



Les fichiers de la politique de sécurité

Exemples de permissions 1/5

- Les classes qui proviennent de l'URL http://pc1.truc.fr/ ont la permission de lire et écrire les fichiers de `/tmp` et le droit de lire les valeurs de toutes les propriétés telles que `user.home`, `user.name`,...

```
grant codeBase "http://pc1.truc.fr/*" {
  permission java.io.FilePermission "/tmp","read";
  permission java.io.FilePermission "/tmp","write";
  permission java.io.FilePermission "/tmp/*","read";
  permission java.io.FilePermission "/tmp/*","write";
  permission java.util.PropertyPermission "user.*","read";
};
```

NB : Il n'est pas possible d'indiquer l'action `"read,write"`

Les fichiers de la politique de sécurité

Exemples de permissions 2/5

- Les classes peuvent se connecter partout par socket et peuvent lire tous les fichiers du répertoire indiqué

```
grant {
  permission java.io.SocketPermission "*","connect";
  permission java.io.FilePermission "C:\\users\\*","read";
};
```

NB : Sous MS-Windows il ne faut pas oublier de doubler les "\"

Les fichiers de la politique de sécurité

Exemples de permissions 3/5

- Permission de lire les fichiers du répertoire `HOME` de l'utilisateur

```
grant {
  permission java.io.FilePermission "${user.home}","read";
  permission java.io.FilePermission "${user.home}${/}*","read";
};
```

Cet exemple utilise la valeur de la propriété système `user.home` (**${user.home}**) et **${/}** qui désigne le séparateur utilisé dans les noms de fichiers (qui dépend du système d'exploitation)

⇒ Les chemins d'accès aux fichiers peuvent être désignés indépendamment de l'OS

Les fichiers de la politique de sécurité

Exemples de permissions 4/5

- Certaines permissions n'ont pas d'action :

```
permission java.lang.RuntimePermission "getClassLoader";
```

- Voire même ni action ni but :

```
permission java.security.AllPermission;
```

  - ex : pour tester du code indépendamment des problèmes d'autorisation
  - **très dangereux !** Ne jamais utiliser un tel fichier de police en production ni même quand l'ordinateur est connecté à Internet

⇒ à n'utiliser qu'avec l'option `-D` de java ; sinon, limiter au moins la portée par un *codeBase*

Les fichiers de la politique de sécurité

Exemples de permissions 5/5

- Si on veut donner l'autorisation d'accéder aux membres non public d'une classe avec la réflexivité, on peut donner la permission suivante :

```
permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
```

  ⇒ C'est évidemment très dangereux et doit être réservé à des classes très particulières comme les débuggeurs ou les outils interactifs pour construire des applications

Les fichiers de la politique de sécurité

Messages d'erreur liés à la sécurité

```
Exception occurred during event dispatching:
  java.security.AccessControlException: access denied
  (java.io.FilePermission C:\ read)
at java.security.AccessControlContext.checkPermission(...)
at java.security.AccessController.checkPermission(...)
at java.lang.SecurityManager.checkPermission(...)
at java.lang.SecurityManager.checkRead(...)
at java.io.File.isDirectory(...)
```

⇒ Indique la permission (type, but, action) qui manque dans les fichiers de la politique de sécurité

- Pour faire afficher les noms des permissions nécessaires à l'exécution, on peut lancer une application avec

  `java -Djava.security.debug=access,failure`

Les fichiers de la politique de sécurité

PropertyPermission

- `PropertyPermission` hérite de `BasicPermission`
- Le "but" d'une telle permission est un nom hiérarchique de type `user.home`, qui peut comporter un `"*"` à la place d'un des noms
- Exemple :

```
permission java.util.PropertyPermission "user.*";
```

donne des permissions sur les propriétés système `user.home`, `user.dir`, `user.name`,...

Les fichiers de la politique de sécurité

FilePermission

- Le "but" peut être de plusieurs types :
  - `fichier` ou `répertoire`
  - `repertoire/*` : fichiers et répertoires situés juste sous `repertoire`
  - `repertoire/-` : fichiers ou répertoires situés dans l'arborescence de `repertoire`
  - `<<ALL FILES>>` : tous les fichiers...
- L'action peut être "`read`", "`write`", "`delete`", "`execute`", mais pas "`read,write`" (2 entrées sont nécessaires)
- **Attention :** "repertoire/*" et "repertoire/-" ne donnent pas la permission sur "`repertoire`" lui-même !

Les fichiers de la politique de sécurité

`FilePermission` : exemple

```
grant {
  permission java.net.FilePermission "/tmp/*","read";
};

grant codeBase "http://gtr.univ-pau.fr/-",signedBy "paul" {
  permission java.net.FilePermission "C:\\users\\foo\\*","write";
};

grant codeBase "http://pc1.truc.fr/*" {
  permission java.io.FilePermission "C:\\temp","read";
  permission java.io.FilePermission "C:\\temp","write";
  permission java.io.FilePermission "C:\\temp\\*","read";
  permission java.io.FilePermission "C:\\temp\\*","write";
};
```

Les fichiers de la politique de sécurité

Entrée `keystore`

- Syntaxe : `keystore "urlFichier" [,"type"];`
- Cette entrée est obligatoire dès qu'une entrée `grant` fait référence à une signature
- Elle doit alors être unique
- Elle peut être n'importe où dans le fichier de politique
- `urlFichier` est une URL absolue ou relative à l'emplacement du fichier de politique de sécurité
- `type` définit le format de stockage et de cryptage des informations contenues dans le fichier
  - → `"jks"` est un type défini par Sun ; c'est le type par défaut

Plan du cours

Concepts pour la cryptographie

Échanger des messages confidentiels

- Propriétés que l'on voudrait garantir :
  - **confidentialité** : seul le destinataire peut lire le message
  - **authentification** : l'expéditeur est bien celui qu'il prétend être
  - **intégrité** : le contenu du message ne peut être modifié par un tiers sans que cela se voit
  - **non-répudiation** : l'expéditeur ne peut pas nier avoir envoyé le message et le destinataire ne peut pas nier l'avoir reçu

Concepts pour la cryptographie

Cryptographie

- Science et techniques pour chiffrer des informations
- Les algorithmes reposent sur la notion de clé
- Une clé est une information utilisée pour chiffrer ou déchiffrer une information
- Deux types principaux chiffrement/déchiffrement :
  - → à clé cachée (ou symétrique)
  - → à clé publique (ou asymétrique)

Concepts pour la cryptographie

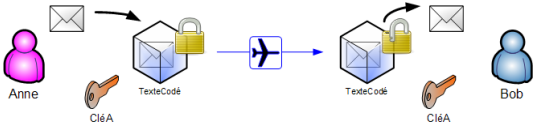
Objectif de cette partie du cours

- Dans ce cours nous n'étudierons pas les algorithmes de chiffrement, ni les chiffrements des messages $\Rightarrow$ mathématiques
- Nous utiliserons la cryptographie en informatique pour :
  - fournir des **certificats**
  - **signer** des documents (ou d'autres informations)
    - $\rightarrow$ l'authentification de l'auteur d'un message
    - $\rightarrow$ l'intégrité du message
    - $\rightarrow$ la non-répudiation de l'auteur du message

Concepts pour la cryptographie

Cryptographie à clé cachée

- C'est la plus ancienne technique de cryptographie
- Une même clé permet de chiffrer et de déchiffrer les messages
- Cette clé doit être partagée par l'expéditeur et le destinataire
- **Problèmes**
  - → Comment s'échanger la clé ?
  - → Si un tiers intercepte la clé pendant l'échange, il peut lire tous les messages et en émettre de nouveaux



Concepts pour la cryptographie

Clé cachée → clé publique

- La cryptographie à clé publique n'a pas le problème de la transmission de la clé
- Les possibilités sont plus riches avec la cryptographie à clé publique
  - → chiffrement (confidentialité)
  - → signature
  - → certificats
- Mais le chiffrement/déchiffrement reste plus rapide avec une clé cachée

Concepts pour la cryptographie

Cryptographie à clé publique

- Chaque acteur (expéditeur ou destinataire) possède deux clés différentes :
  - une **clé publique** connue de tous
  - une **clé privée** connue de lui seul
- Ces 2 clés sont "liées" $\Rightarrow$ un message chiffré avec une des 2 clés ne peut être déchiffré qu'avec l'autre clé
- Cette solution évite le problème d'échange de clé symétrique et offre de nouvelles possibilités

Concepts pour la cryptographie

Cryptographie à clé publique : principe essentiel

- Les 2 clés sont générées par des algorithmes qui s'appuient sur des théories mathématiques (arithmétique des grands nombres premiers) qui leur assurent la propriété suivante :
  - il est facile de chiffrer un message avec la clé publique
  - il est **extrêmement difficile** de déchiffrer le message si on ne connaît pas la clé privée associée

Concepts pour la cryptographie

Infrastructure à clé publique

- L'utilisation des clés publiques/privées impose la gestion et la mise à disposition des clés publiques ainsi que la certification des identités associées à ces clés
- Des tiers effectuent ces tâches : autorités d'enregistrement et de certification
- Pour travailler à grande échelle avec les clés publiques il est donc nécessaire d'installer une infrastructure à clé publique (ICP en abrégé, PKI en anglais)

Concepts pour la cryptographie

Chiffrement avec une clé publique

- Envoi d'un message chiffré :
  1. l'expéditeur **chiffre son message avec la clé publique du destinataire** (elle est connue de tous)
  2. le destinataire **déchiffre avec sa clé privée** (restée chez lui "bien en sécurité")
- On assure ainsi :
  - la confidentialité
  - l'authentification du destinataire
- Pour l'intégrité, l'authentification et la non répudiation de l'expéditeur, il faut ajouter une signature

Plan du cours

Signatures numériques

Chiffrement avec une clé publique

- Un message peut être chiffré avec la **clé privée de l'émetteur** ; ce message pourra être déchiffré (par tout le monde) avec la clé publique correspondante

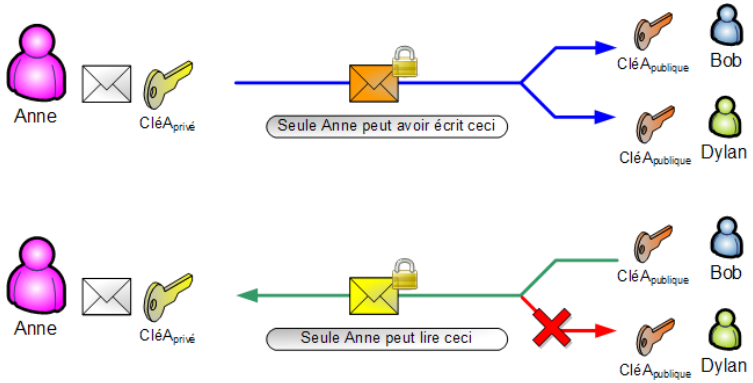
⇒ Ce sens sert pour les **signatures digitales** (intégrité)

- Un message peut être chiffré avec la **clé publique du destinataire** ; ce message ne pourra être déchiffré que par le destinataire (avec la clé privée correspondate)

⇒ Ceci sert à transmettre des **messages secrets** (confidentialité)

Signatures numériques

Chiffrement avec une clé publique



Signatures numériques

Propriétés des signatures à clés asymétriques

- Une signature numérique à clé publique permet :
  - **authentification** : l'expéditeur est bien celui qu'il prétend être
  - **intégrité** : les données n'ont pas été modifiées depuis que le document a été signé
  - **non répudiation** : l'expéditeur ne peut pas nier avoir signé le document
- Une telle signature permet d'affirmer que le signataire du message possède bien une certaine clé publique
- Cette signature ne peut pas être imitée si on ne connaît pas la clé privée du signataire

Signatures numériques

Notion de résumé

- Un résumé (ou empreinte, condensé, *message digest*) est une information de taille fixe calculée à partir du contenu de message par une fonction de hachage
- La taille du résumé est bien plus petite que la taille du message $\Rightarrow$ gain de temps lors du chiffrement
- Cette fonction de hachage $h : T \rightarrow R$ associe à tout texte T un résumé R de longueur fixe ; cette fonction doit avoir les propriétés suivantes :
  - il est très facile de calculer R
  - il est très difficile (voire impossible) de calculer T à partir de R
  - il est très difficile de trouver T' tel que $h(T') = h(T)$
  - h est "presque bijective", i.e. si $T_1 \neq T_2$ alors il est "presque certain" que $h(T_1) \neq h(T_2)$

Signatures numériques

Fonctions de hachage

- Fonctions les plus couramment utilisées :
  - MD4 et **MD5** (*Message Digest*) furent développées par Ron Rivest ; MD5 produit des hachés de 128 bits en travaillant les données originales par blocs de 512 bits
  - **SHA-1** (*Secure Hash Algorithm 1*), comme MD5, est basé sur MD4 ; il fonctionne également à partir de blocs de 512 bits de données et produit par contre des condensés de 160 bits en sortie ; il nécessite donc plus de ressources que MD5
  - SHA-2 (*Secure Hash Algorithm 2*) a été publié récemment ; différences tailles possibles pour les hachés : 256, 384 ou 512 bits ; il sera bientôt la nouvelle référence en termes de fonction de hachage
  - RIPEMD-160 (*Ripe Message Digest*) est la dernière version de l'algorithme RIPEMD ; elle produit des condensés de 160 bits (128 bits pour la version précédente, mais failles de sécurité importantes) ; elle demande plus de ressources que SHA-1 (son principal concurrent)

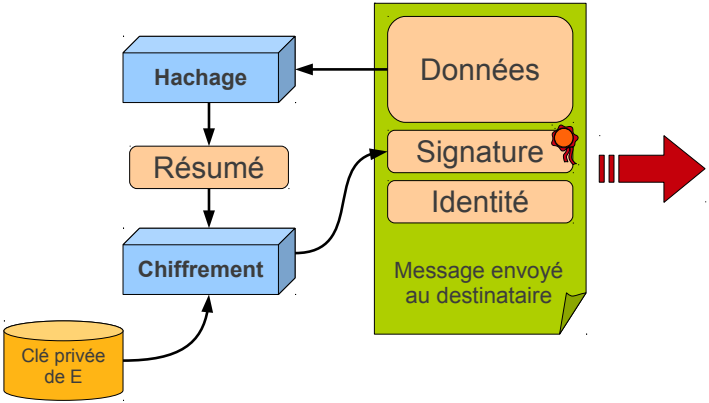
Signatures numériques

Comment signer ?

- Une signature est composée en chiffrant un résumé du message avec la clé privée du signataire
- Le destinataire va déchiffrer le résumé avec la clé publique du signataire
- Si le résultat correspond bien au résumé du message reçu, cela signifie que :
  - le signataire est le bon (celui qui possède cette clé publique)
  - le message n'a pas été modifié depuis le calcul de la signature

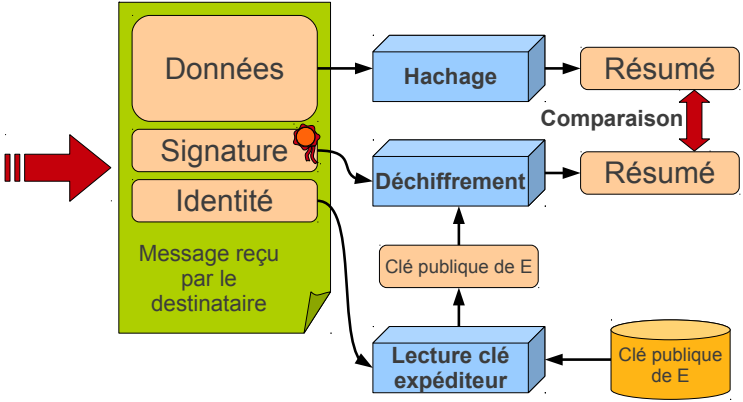
Signatures numériques

Signature : envoi des données



Signatures numériques

Signature : réception des données



Signatures numériques

Robustesse ?

- Le MD5 a été "cassé"
  - ⇒ Il est possible de modifier un document de manière à ce que le MD5 du document modifié soit identique au MD5 du document initial !
  - Pb : cela nécessite généralement de "grosses" modifications du document
  - Solution : "cacher" tout ou partie de ces modifications dans des champs cachés :
    - → commentaires dans un `.doc` ou `.html`
    - → informations non imprimables dans un `.pdf`

Signatures numériques

Robustesse ?

- Comment utiliser cette faille ?
  - Soit T le document initial ; sa signature est basée sur $h(T)$
  - En modifiant T on obtient un nouveau document T'
  - Problème : $h(T') \neq h(T)$
  - Solution : générer un nouveau document $T" = T' + \Delta$ tel que $h(T") = h(T)$
  - Objectif : connaissant T' et $h(T)$, trouver Δ

$\Rightarrow$ Puisque la fonction de hachage calcule le même résumé pour T et $T"$, la signature apposée sur T sera également valide sur $T"$

$\Rightarrow$ Il n'est même pas nécessaire de connaître la clé privée qui a servi à chiffrer le résumé !

Signatures numériques

Robustesse ?

- Fatalement, tôt ou tard, les autres algos de hachage subiront le même sort !
- Solution :
  - Fournir 2 résumés au lieu d'un seul ⇒ **2 signatures**
  - On suppose que modifier le document pour que les 2 résumés restent identiques implique de si "lourdes" modifications que cela est quasi impossible... pour le moment...

⇒ Solution mise en œuvre par SUN : MD5 + SHA1

Plan du cours

Certificats

Association clés-identités

- Celui qui reçoit un message signé doit posséder la clé publique du signataire et avoir un moyen d'associer cette clé publique à l'identité du signataire (base de données ou autre)

⇒ Imaginez le travail pour :

- gérer les milliers de clés publiques des clients et des partenaires
- s'assurer que ces clés appartiennent bien à ces correspondants

⇒ Dans la réalité

- celui qui reçoit le message n'a pas la clé de l'expéditeur
- la clé publique du signataire est transmise avec le message, dans un **certificat** qui associe cette clé avec l'identité du signataire

Certificats

Comment avoir confiance en ce certificat ?

- Le certificat est signé par une **autorité de certification** (AC) publique (comme Verisign) qui a une clé publique bien connue de tous (ou il existe des moyens sûrs et faciles de l'obtenir)

⇒ On se retrouve dans le cas où on connaît la clé de l'expéditeur des données (la donnée est le certificat) ; on peut donc s'assurer que le certificat contient des données exactes : association clé publique-identité

⇒ On a ainsi la clé publique de l'expéditeur ⇒ on peut donc vérifier la signature des données, etc...

Certificats

Contenu d'un certificat

- L'AC fournit un certificat **signé** par elle, qui contient :
  - des informations sur le possesseur certificat
    → *nom, adresse, etc...*
  - la **clé publique** du possesseur
  - des informations liées au certificat
    → *date d'émission, date de validité, numéro de série, etc...*
- Les normes :
  - **OpenPGP** [18] (RFC 4880) : format pour l'échange sécurisé de données
  - norme **X.509** : un certificat contient des champs d'information certifiés par des autorités de certification reconnues

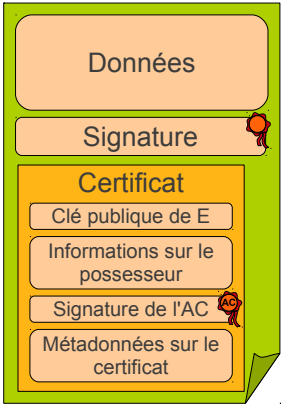
18. PGP = Pretty Good Privacy

Certificats

Contenu d'un certificat

- Un certificat est un message signé

⇒ Si on connaît la clé publique de l'autorité, et si on a confiance en cette autorité, alors on peut être sûr des informations qu'il contient.



Certificats

La chaîne de vérification

- Le vérificateur du certificat doit déjà connaître la clé publique de (appelons-la CA_1) l'autorité de certification
- Sinon, le vérificateur doit posséder un certificat, émis par une autorité de certification CA_2 dont il connaît la clé publique, et qui authentifie la clé publique de CA_1
- On peut ainsi avoir une chaîne de certificats signés par CA_1, CA_2, CA_3,...
- Cette chaîne confirme l'identité du signataire si un des certificats a été signé par une autorité déjà connue par le vérificateur

Certificats

Obtention d'un certificat

1. Génération d'une paire de clés publique-privée
2. Fournir à l'autorité qui délivre le certificat
   - la clé publique
   - des documents qui certifient l'identité
   - le réglement des frais de certification...
3. L'autorité fournit un certificat **signé** par elle, qui permet d'associer la clé publique et l'identité

⇒ Pour les cas de diffusion restreinte, on peut se contenter d'un certificat auto-signé que l'on distribue aux utilisateurs

Certificats

Obtention d'un certificat

- Les autorités universellement connues ne sont pas les seules à délivrer des certificats
- Un organisme peut aussi utiliser **en interne** sa propre autorité de certification
- Un certificat peut aussi être **auto-signé**, c'est-à- dire signé par la personne qui est authentifiée par le certificat (convient si le destinataire connaît la clé publique de celui qui a signé)

Certificats

Stockage des clés & certificats en Java

- Toute application Java peut avoir un lieu de stockage des clés publiques
- Cette base de données contient des clés publiques et des chaînes de certificats qui authentifient les clés publiques
- Elle peut aussi contenir des clés privées locales (celles de l'utilisateur de l'application) avec leurs chaînes de certificats
- Des mots de passe peuvent être associés au lieu de stockage et à chacune des clés

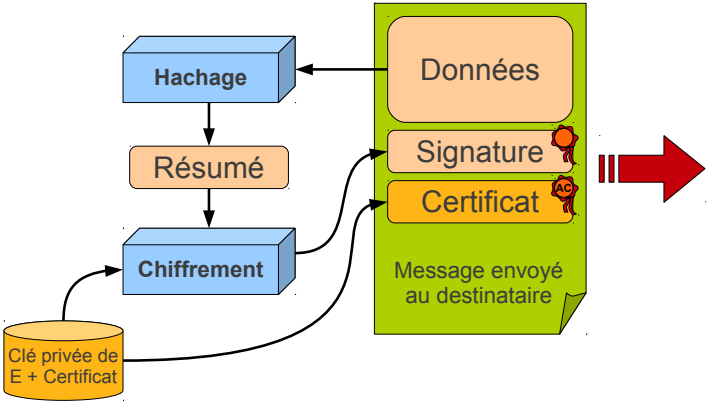
Certificats

Entrepôt de certificats par défaut pour Java

- Par défaut les clés et certificats sont entreposés dans le fichier `java.home/lib/security/`
  (`java.home` est le répertoire `jre` d'installation de Java)
- Pour lister les certificats : `keytool -list`
- On peut indiquer un autre fichier entrepôt :
  `keytool -list -file entrepot`

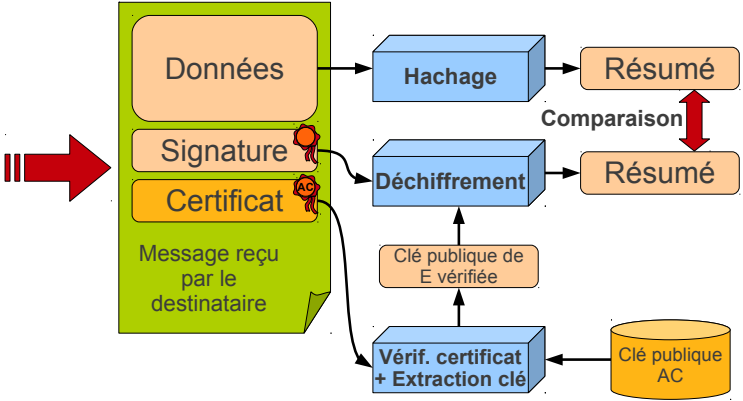
Certificats

Envoi des données signées



Certificats

Réception des données signées



Plan du cours

Outils de SUN pour la signature

Outils dédiés à la sécurité

`keytool` permet de gérer les clés et les certificats

`policytool` facilite la saisie d'une politique de sécurité (évite les fautes de syntaxe que l'on peut faire si on modifie "à la main" les fichiers de la politique)

`jarsigner` permet de signer un fichier `.jar` ou de vérifier sa signature

Outils de SUN pour la signature

Algorithmes utilisés dans les API Java

- Fonction de hachage par défaut : `SHA1`
- Chiffrement par défaut pour les signatures : `DSA`

⇒ Ce qui donne l'algorithme de signature de nom interne `SHA1withDSA`

- Autres possibilités : `MD2withRSA`, `MD5withRSA`, `SHA1withRSA`
- Certificats selon la norme `X.509`

Outils de SUN pour la signature

Exemple : création des clés & certificats

- Création d'une paire de clés (privée et publique) de `toto`, et d'un certificat **auto-signé** valable 90 jours (rangés dans le fichier `.keystore`) :

```
> keytool -genkey -alias toto -keystore .keystore
Enter keystore password: abc123
What is your first and last name?
  [Unknown]: Pierre Toto
What is the name of your organizational unit?
  [Unknown]: DptInformatique
What is the name of your organization?
  [Unknown]: UPPA
What is the name of your City or Locality?
  [Unknown]: Pau
What is the name of your State or Province?
  [Unknown]: PA
What is the two-letter country code for this unit?
  [Unknown]: FR
Is <CN=Pierre Toto, OU=DptInformatique, O=UPPA,
    L=Pau, ST=PA, C=FR> correct?
  [no]: y
```

Outils de SUN pour la signature

Exemple : obtention d'un certificat validé

- Objectif : obtenir un certificat (ou une chaîne de certificats) pour certifier la clé publique de `toto`, émis par une autorité publique connue
  - envoyer la clé publique et les documents demandés par l'autorité pour identifier `toto`
  - remplacer le certificat auto-signé de `toto` par ce nouveau certificat (reçu dans le fichier `reponseCA`), dans le fichier `.keystore` de `toto` :

```
> keytool -import -alias toto -keystore .keystore -file reponseCA
```

⇒ cette étape n'est pas obligatoire ; on peut se contenter du certificat auto-signé

Outils de SUN pour la signature

Exemple : exporter le certificat

- Exporter le certificat de `toto` et l'enregistrer dans le fichier `toto.crt`

```
> keytool -export -alias toto -keystore .keystore -file toto.cer
```

- Ce certificat pourra être envoyé pour être importé dans la base de données des clés de celui qui fera confiance à `toto`
- NB : Bien évidemment, ce certificat ne contiendra que la clé publique de `toto`

Outils de SUN pour la signature

Exemple : signer une archive `.jar`

- Créer le fichier `unjar.jar` normallement :

```
> jar cvf unjar.jar Classe.class ...
```

- `toto` signe cette archive `unjar.jar` et obtient ainsi le fichier `unjarsigne.jar` :

```
> jarsigner -keystore .keystore -signedjar unjarsigne.jar unjar.jar toto
```

  → cette commande demandera à l'utilisateur de saisir le mot de passe pour accéder au keystore afin de récupérer la clé privée de `toto` pour signer l'archive

Outils de SUN pour la signature

Exemple : fichier MANIFEST.MF d'un .jar signé

```
Manifest-Version: 1.0
Created-By: 1.6.0_26 (Sun Microsystems Inc.)
Main-Class: main.Main

Name: main/Truc.class
SHA1-Digest: yIPcOrlW7SE5qkBFvi738cjUpaQ=

Name: main/Main.class
SHA1-Digest: rw9e3vEoYBBCHHKE6OGmquPhoRY=
```

Outils de SUN pour la signature

Exemple : répertoire `META-INF` d'un `.jar` signé

- Fichiers supplémentaires dans un `.jar` signé par `alice` :
  - → fichier `ALICE.SF`

```
Signature-Version: 1.0
SHA1-Digest-Manifest-Main-Attributes: hc4dQqVKBLuV1+Mvn2s/HwjeCFU=
Created-By: 1.6.0_26 (Sun Microsystems Inc.)
SHA1-Digest-Manifest: TEibACFibjBIB5qL5VRBexpbK0k=

Name: main/Truc.class
SHA1-Digest: Vhoe3xVn6YsYL35VochJNQk343w=

Name: main/Main.class
SHA1-Digest: wDncTm1P+dFPQQBTwI/nG+DHucs=
```

  - → fichier `ALICE.RSA` : ???

Normalement, une clé RSA est au format texte...

Outils de SUN pour la signature

Exemple : à la réception d'un `.jar` signé

- L'utilisateur qui recevra de `toto` le fichier signé va vérifier que c'est bien `toto` qui l'a signé
  1. il vérifie d'abord que le certificat de `toto` qu'on lui a envoyé contient des informations qui correspondent aux informations qu'il a reçu par d'autres voies fiables
  2. il doit ensuite importer le certificat de `toto` dans sa base de certificats
  3. il peut finalement utiliser cette base de clés en donnant son emplacement dans le fichier de politique de sécurité (entrée `keystore`)

Outils de SUN pour la signature

Exemple : importer un certificat

- Vérifier le certificat en comparant les informations imprimées par `keytool` avec des informations fiables obtenues par ailleurs :

```
> keytool -printcert -file toto.cer
```

- Importer le certificat de `toto`, utilisateur en qui on peut avoir confiance :

```
> keytool -import -alias toto -file toto.cer -keystore .keystore
```

Outils de SUN pour la signature

Certificats révoqués

- Java ne vérifie pas auprès de l'autorité de certification si le certificat a été révoqué (en cas, par exemple, de vol ou de certificat attribué par erreur)
- Pour les cas où une sécurité renforcée est nécessaire, il faudra ajouter du code pour effectuer cette vérification "à la main"

⇒ CRL = *Certificate Revocation List*

Outils de SUN pour la signature

Signature & plugin Java

- Depuis la version JDK 1.3, une applet signée et certifiée peut avoir tous les droits si l'utilisateur déclare avoir confiance dans cette applet signée (le navigateur lui pose la question)
- Cette fonctionnalité a été ajoutée pour faciliter l'exécution des applets sans installation de fichiers de politique de sécurité spéciaux chez les clients Web

Outils de SUN pour la signature

Signature & plugin Java

- Mais cette possibilité peut être jugée dangereuse !
  - ⇒ Le fichier de politique de sécurité du client peut comporter la ligne suivante

    ```
    permission java.lang.RuntimePermission "usePolicy";
    ```

    pour indiquer que seule la politique de sécurité doit être examinée, et que même une applet signée devra s'y tenir

Outils de SUN pour la signature

Signature & plugin Java

- Le plugin Java a changé de politique de sécurité à chaque version du JRE, ce qui n'a pas favorisé la portabilité
- Les fonctionnalités ont été ajoutées pour faciliter le déploiement des applets
  - avec le JRE 1.2, seuls les fichiers de politique de sécurité sont pris en compte
  - avec le JRE 1.3, il n'est pas tenu compte de ces fichiers de politique si l'applet est signée avec un certificat obtenu par une autorité connue de certification, et si l'utilisateur affirme avoir confiance en ce certificat
  - depuis le JRE 1.4, ça fonctionne aussi avec les certificats auto-signés

Plan du cours

Code privilégié

- Une portion de code peut recevoir des droits privilégiés s'il est exécuté par la méthode `static doPrivileged()` (classe `AccessController`)
- Cette méthode prend en paramètre une instance d'une classe qui implémente l'interface `PrivilegedAction`
- Cette interface ne comprend qu'une méthode `Object run()`
- La méthode `doPrivileged` exécute cette méthode `run()` et retourne l'objet renvoyé par `run()`

Code privilégié

- Les autorisations d'accès du code privilégié (exécuté par la méthode `doPrivileged`) dépendent uniquement des droits :
  - du code privilégié
  - des méthodes appelées par ce code
- Il y a donc moins de vérifications à faire : **les limitations liées aux méthodes qui ont appelé ce code n'interviennent plus**
- Soyez donc très prudent avec ce que vous permettez dans du code privilégié ! ! !

Code privilégié

Comment lever des exceptions ?

- La méthode `run()` ne peut pas renvoyer d'exception
- Sinon, on doit utiliser l'interface `PrivilegedExceptionAction` dont la méthode `run()` renvoie une `Exception`

Code privilégié

Permissions nécessaires

- Aucune permission spéciale n'est requise pour exécuter du code privilégié
- Ça n'est pas nécessaire car le code privilégié ne permet pas d'exécuter du code qui ne serait pas autorisé normalement par la classe

Code privilégié

Exemple de code privilégié

- Code extrait du constructeur de la classe `java.io.PrintWriter` :

```
lineSeparator=(String)AccessController.doPrivileged(
  new sun.security.action.GetPropertyAction("line.separator"));
```

- La variable `lineSeparator` est utilisée par le code de la méthode `println()`
- `doPrivileged` assure que `line.separator` pourra être lue quelle que soit la politique de sécurité et les méthodes qui ont appelé ce code

Code privilégié

Autre exemple de code privilégié

```
package truc;

import java.security.AccessController;
import java.security.PrivilegedAction;
import java.io.File;

public class Truc {
  public static void test() {
    AccessController.doPrivileged(new PrivilegedAction() {
      public Object run() {
        File f=new File("/tmp/my-test-file");
        f.delete();
        return null;
      }
    });
  }
}
```

Code privilégié

Exemple de problème à résoudre 1/2

- **Contexte :** On a un fichier sensible `password` dans lequel il faut absolument respecter un format spécial
- **Problème :** Comment permettre l'écriture dans ce fichier sans permettre l'écriture de lignes qui ne respectent pas ce format ?
- **Solution :** Écrire une classe spéciale, seule autorisée à écrire dans le fichier (utiliser son *codeBase* dans le fichier de police), comportant une méthode `ecrire` qui effectue les écritures dans le fichier `password` **en mode privilégié**
  - les autres classes lui délèguent les écritures dans `password`
  - elles pourront ainsi enregistrer dans le fichier sans risque
  - sans le mode privilégié, elles n'auraient pas eu le droit d'écrire dans le fichier

Code privilégié

Exemple de problème à résoudre 2/2

```
static void ecrire(final String s) throws IOException {
  if (pas bon format)
    throw new IOException("Mauvais format");
  try {
    AccessController.doPrivileged(
      new PrivilegedExceptionAction() {
        ...
        public Object run() throws IOException {
          // Ecriture de s dans le fichier password
          ...
        }
      });
    } catch(PrivilegedException e) {
      // Renvoie l'exception levee par run()
      throw e.getException();
  }
}
```

Contexte d'appel & threads

- Quand un nouveau *thread* est créé, le contexte pour le contrôle d'accès est hérité par ce *thread*
- L'héritage se fait au moment de la création et pas du lancement du thread

Protéger des objets

- On peut protéger un objet particulier

- `GuardedObject` permet de faire garder un objet par un autre objet qui est d'une classe qui implémente l'interface `Guard` :

```
GuardedObject go = new GuardedObject(
    new ClasseProtegee(...),
    new GardeDuCorps(...)
);
```

NB : la classe `GardeDuCorps` doit implémenter l'interface `Guard`

Protéger des objets

Interface `Guard`

- Elle contient seulement la méthode `checkGuard()` qui lève une `SecurityException` si l'accès à l'objet est interdit

- Cette interface est implémentée par les classes filles de la classe `java.Security.Permissi`
  - ⇒ Dans ce cas, la méthode `checkGuard()` fait simplement un appel à `securityManager.checkPermission(this)`

Protéger des objets

Exemple

- Si on garde un objet avec une instance de `FilePermission`, alors l'accès à l'objet sera autorisé dans les mêmes conditions que l'accès à un fichier donné

- Avec le code suivant, `objet` ne pourra être accéder que par le code qui aura l'autorisation de lire le fichier `/rep1/fichier` :

```
GuardedObject go = new GuardedObject(objet,
    new FilePermission("/rep1/fichier","read")
);
```

NB : Il n'est même pas nécessaire que le fichier `/rep1/fichier` existe réellement sur le *file system*

Protéger des objets

Récupération de l'objet protégé

- Quand on veut récupérer l'objet protégé, on appelle la méthode `getObject()` de la classe `GuardedObject`
- `getObject()` fait automatiquement un appel à la méthode `checkGuard()` qui lèvera une `SecurityException` pour signaler que l'accès à l'objet est interdit

Protéger les paquetages

- Par défaut, les paquetages ne sont pas protégés
- Un programmeur peut ajouter une classe dans n'importe quel paquetage et avoir ainsi accès à tous les membres des classes du paquetage qui ont un accès réservé au paquetage
- On peut empêcher l'ajout de nouvelles classes dans les paquetages par divers moyens :
  - par les mécanismes de protection liés aux fichiers de politique de sécurité
  - par les fichiers `.jar` **scellés**

Protéger les paquetages

Politique de sécurité & permissions

- Via les fichiers de sécurité de Java, on peut empêcher :
  - le chargement direct d'une classe d'un paquetage
  - l'ajout de nouvelles classes dans un paquetage
- Il faut pour cela ajouter des entrées dans le fichier de sécurité (`java.security`)
  - ces entrées sont des listes de noms séparés par des `","`
  - tout paquetage dont le nom **commence** ainsi est protégé

→ pour empêcher le chargement direct :

```
package.access=fr.univ-pau.librairie,com.truc
```

→ pour empêcher l'ajout de nouvelles classes :

```
package.definition=fr.univ-pau,com.truc
```

Protéger les paquetages

Politique de sécurité & permissions

- Si on a protégé un paquetage, on peut autoriser l'action interdite en ajoutant des entrées dans les fichiers de politique de sécurité (`.policy`)
- L'autorisation suivante permettra un accès direct aux classes du paquetage `fr.univ-pau.t`

  `RuntimePermission "accessClassInPackage.fr.univ-pau.toto.truc"`
- De même, l'autorisation suivante permettra l'ajout d'une classe dans un paquetage

  `RuntimePermission "defineClassInPackage.`⟨*pkgName*⟩`"`

Protéger les paquetages

Chargeurs de classes

- La protection des paquetage n'est pas vraiment prise en compte par les chargeurs de classes jusqu'à la version 1.3
- La protection d'accès n'est prise en compte que par la classe `URLClassLoader`, et encore, seulement si l'instance du chargeur de classes a été obtenue par la méthode `static newInstance()` et non par un `new`
- La protection pour l'ajout d'une classe dans un paquetage n'est prise en compte par aucun chargeur de classes

Protéger les paquetages

Fichiers .jar scellés

- L'entrée "`Sealed: true`" dans la section principale du fichier `MANIFEST` d'un `.jar` empêche l'ajout, dans les paquetages de cette archive, de classes venant d'une autre source que ce fichier `.jar`
- On peut aussi ne protéger que certains paquetages du fichier `.jar` en donnant des entrées du type

```
Name: fr/univ-pau/p1
Sealed: true
```

Plan du cours

Java Authentification and Authorization Service

- Le mécanisme des autorisations de Java 2 est centré sur le code : les permissions sont données ou non selon l'origine du code (d'où vient-il, qui l'a signé)
- JAAS est **centré sur l'utilisateur** : il va permettre de donner des permissions en se basant sur l'utilisateur qui exécute le code
- Les utilisateurs devront s'authentifier et certaines parties du code ne pourront être exécutées que par les utilisateurs autorisés

Java Authentification and Authorization Service

- Concepts de base de JAAS :

subject représente une entité authentifiée ; utilisateur, administrateur, service web, processus,...

principal une des identités d'un sujet ; un sujet peut en avoir plusieurs, sur le modèle d'un utilisateur qui peut avoir plusieurs noms pour des services différents, un numéro de sécurité sociale, une adresse mail,...

credential justificatif ("pièce d'identité") pour une identité ; peut être un objet quelconque

Java Authentification and Authorization Service

- Depuis la version 1.4, JAAS est inclus dans le J2SE
  - `javax.security.auth` contient la classe `Subject`
  - `javax.security.auth.login` contient la classe `LoginContext`
  - `javax.security.auth.spi` contient la classe `LoginModule`
  - `javax.security.auth.callback` contient les interfaces `Callback` et `CallbackHandler`, ainsi que quelques classes qui implémentent `Callback`

Java Authentification and Authorization Service

- Il y a 2 parties dans JAAS :
  1. L'**authentification** de l'utilisateur (ex : avec un système de login et mot de passe)
  2. Les **autorisations**, liées à l'utilisateur qui s'est authentifié, qui s'ajoutent au système des autorisations de Java 2 déjà étudié
- Exemple minimal de code

```
LoginContext lc = new LoginContext("Appli1");
try {
    lc.login();
}
catch(LoginException e) {
    // Le Login n'a pas fonctionne
}
Subject sujet = lc.getSubject();
Subject.doAs(sujet, new ActionProtegee());
```

JAAS : authentification

- L'architecture JAAS a pour objectif de découpler :
  - les **informations** à demander à l'utilisateur
    → *login, password,...*
  - la **manière de demander** ces informations à l'utilisateur
    → *interface Swing, mode textuel, carte à puce,...*
  - les **mécanismes d'authentification** utilisés pour valider (ou non) la connexion de l'utilisateur
    → *fichier login/password, serveur LDAP, Kerberos,...*

JAAS : authentification

- Elle est effectuée par une instance de la classe `LoginContext`
- Quand on crée cette instance, on lui passe un nom qui va permettre à JAAS de savoir comment se fera l'authentification de l'utilisateur
  - → ce nom désigne une entrée dans un fichier de configuration
  - → créer un `LoginContext` nécessite la permission

    `javax.security.auth.AuthPermission "createLoginContext"`
- Emplacement du fichier de configuration de JAAS :
  - via la propriété Java

    `java.security.auth.login.config`
  - via une ou plusieurs entrées dans le fichier `java.security`

    `login.config.url.1=file:${java.home}/lib/security/truc.login`

JAAS : authentification

- Le fichier de configuration contient des entrées qui indiquent quels modules de login devront être utilisés :

```
entree1 {
    ClasseModule1 flag options;
    ClasseModule2 flag options;
    ...
}
entree2 {...}
```

- Exemple :

```
Appli1 {
    fr.univ-pau.SgbdLoginModule Required
        driver="org.gjt.mm.mysql.Driver"
        url="jdbc:mysql://m.univ-pau.fr/jaas?user=adm"
        debug="true";
}
Appli2 {
    fr.univ-pau.FichierLoginModule Required
        fichier="rep/motdepasse";
}
```

JAAS : authentification

Système de plugins

- La classe `LoginContext` utilise des "plugins", appelés **modules de login**, pour réaliser l'authentification de l'utilisateur
- Le paquetage `com.sun.security.auth.module` contient des modules JAAS pour authentifier avec :
  - nom et mot de passe Unix (`UnixLoginModule`) ou NT (`NTLoginModule`)
  - un service JNDI (`JndiLoginModule`)
  - le protocole Kerberos (`Kbr5LoginModule`)
  - une clé enregistrée dans un fichier KeyStore (`KeyStoreLoginModule`)
  - un annuaire LDAP (`LdapLoginModule`)
- Il est également possible d'écrire ses propres modules de login ou de les acheter

JAAS : authentification

Callbacks & callback handlers

- Pour accroître la portabilité des applications, la méthode `login` échange des informations avec l'utilisateur à authentifier par l'intermédiaire d'un `CallbackHandler`

⇒ Ainsi, en changeant ainsi de *handler*, on pourra demander le nom et le mot de passe d'un utilisateur en mode texte dans une console, ou dans une fenêtre Swing, ou par tout autre moyen, sans changer le code de la méthode `login`

JAAS : authentification

Callbacks & callback handlers

- Les échanges avec l'utilisateur sont déterminés par un tableau d'objets `Callback` qui sera passé au `CallbackHandler`
- Un `Callback` n'a pas d'action directe avec l'utilisateur ; il sert seulement à passer les informations échangées entre l'application et l'utilisateur
- C'est le `CallbackHandler` qui va, en s'aidant des `Callback`, afficher des informations à l'utilisateur et lui demander de saisir les données

JAAS : authentification

Callbacks

- Le plus souvent le développeur utilisera les classes de *callbacks* fournies avec l'API (ou avec un module de login JAAS)

⇒ Paquetage `javax.security.auth.callback`
  - `NameCallback`,
  - `PasswordCallback`,
  - `TextOutputCallback`,
  - `ChoiceCallback`,...

- Il est possible d'écrire ses propres *callbacks*

JAAS : authentification

Callback handlers

- Il existe deux `CallbackHandler` dans l'API de base dans le paquetage `com.sun.security.auth.callback`
  - `DialogCallbackHandler` → fenêtre dialogue Swing
  - `TextCallbackHandler` → interface textuelle dans une console
- Le développeur peut bien évidemment écrire son propre *callback handler* pour indiquer comment demander des informations à l'utilisateur à l'aide des *callbacks*
  - ⇒ il doit implémenter l'interface `CallbackHandler` qui possède une méthode : `void handle(Callback[])`

JAAS : authentification

Utilisation des *callbacks*

1. La méthode `login` crée les différents `Callback` nécessaires aux interactions
   → ex : un `NameCallback` et un `PasswordCallback`
2. Elle exécute alors la méthode `handle` du *handler* en lui passant les *callbacks* dans un tableau
3. Le *handler* détermine les interactions qu'il doit faire avec l'utilisateur en fonction du type de chaque *callback*
   → il peut interroger un *callback* (méthodes `getXXX`) s'il veut des précisions (ex : pour obtenir un texte à afficher à l'utilisateur)

JAAS : authentification

Utilisation des *callbacks*

4. Le *handler* met dans les *callbacks* les informations recueillies auprès de l'utilisateur (méthodes `setXXX`)
5. La méthode `login` récupère ces informations avec les méthodes `getXXX` de la classe du *callback*

⇒ Elle peut maintenant les envoyer au module de login pour réaliser l'authentification

JAAS : authentification

Exemple de *callback*

- `NameCallback` implémente l'interface "marqueur" `Callback`
- Cette classe a 2 constructeurs (cf. méthode `login`) :
  - `NameCallback(String prompt)`
  - `NameCallback(String prompt,String nomParDefaut)`
- Cette classe possède les méthodes suivantes :
  - `getPrompt()`
  - `getName()`
  - `setName(String nom)`
  - `getDefaultName()`

JAAS : autorisations

- Une fois que l'utilisateur s'est authentifié, on peut récupérer une instance de `Subject` par `loginContext.getSubject()`
- On peut ensuite accorder des permissions à des actions selon l'utilisateur qui s'est authentifié
- L'exécution de ces actions devra être lancée par la méthode statique `doAs` de la classe `Subject`

```
Subject sujet = lc.getSubject();
ActionSpeciale action = new ActionSpeciale();
Subject.doAs(subject,action);
```

JAAS : autorisations

Interface `PrivilegedAction`

- Les actions passées à la méthode `doAs` doivent appartenir à une classe qui implémente l'interface `PrivilegedAction`
- Cette interface définit la méthode `Object run()` qui contiendra le code de l'action à exécuter
- Le résultat de la méthode `doAs` est la référence `Object` renvoyée par `run`
- Comme avec la méthode `doPrivileged` étudiée précédemment (cf. code privilégié), une action qui veut lever des exceptions doit implémenter `PrivilegedExceptionAction`

JAAS : autorisations

Mode privilégié

- La méthode `doAsPrivileged` permet de lancer l'action dans un autre contexte d'exécution (en mode privilégié si on lui passe `null` comme contexte)

⇒ Attention à ne pas ouvrir des portes aux pirates !

- Les classes qui appellent la méthode `doAsPrivileged` doivent avoir la permission

  `javax.security.auth.AuthPermission "doAsPrivileged"`

JAAS : autorisations

Attribuer des permissions

- On peut maintenant ajouter des informations relatives à JAAS dans les fichiers de politique de sécurité
- Syntaxe :

```
grant codebase ..., signed by ...,
    Principal classePrincipal utilisateur
{
    permission ...;
    permission ...;
}
```

- Exemple :

```
grant codebase "http://..."
    Principal fr.truc.Principal1 "pierre"
{
    permission java.util.PropertyPermission "user.home","read";
    ...
}
```

Plan du cours

Gestion des risques liés à la sécurité de l'information

- L'enjeu : atteindre ses objectifs (de sécurité) sur la base de décisions rationnelles
  - *Née dans le domaine financier dans les années 50 et étendue à de nombreux autres domaines tels que la gestion de projet, la sécurité des personnes, la sûreté de fonctionnement, le marketing, l'environnement ou encore la sécurité de l'information, la* ***gestion des risques*** *a toujours eu pour objectif de rationaliser des situations pour aider à une prise de décision éclairée.*
  - *Les choix effectués par les décideurs peuvent ainsi être faits au regard des éléments fournis par les* ***risk managers****. Et ces choix peuvent autant guider l'organisme vers l'atteinte de ses objectifs que faire évoluer sa stratégie.*

Gestion des risques liés à la sécurité de l'information

- Gestion des risques par la pratique
  - → méthodologie empirique
  - → sources de désaccords

- Gestion des risques par la théorie
  - → formalisme & organisation
  - → différentes normes & méthodes
    - EBIOS 2010
    - suite ISO/CEI 27000
      notamment ISO/CEI 27005:2011 Risk Manager
  - → processus d'audit & de certification

Gestion des risques liés à la sécurité de l'information

- Des pratiques différentes mais des principes communs
  - le risque est décrit par un événement, ses conséquences et sa vraisemblance
  - le processus de gestion des risques comprend une étude du contexte, l'appréciation des risques, le traitement des risques, la validation du traitement des risques, la communication relative aux risques, le contrôle, dans une amélioration continue
- Le besoin d'une méthode
  - disposer d'éléments de langage communs
  - disposer d'une démarche claire et structurée à respecter
  - se baser sur un référentiel validé par l'expérience
  - s'assurer d'une exhaustivité des actions à entreprendre
  - réutiliser la même approche en amélioration continue et sur d'autres périmètres...

Gestion des risques liés à la sécurité de l'information

Suite ISO/CEI 27000

- 1ère norme de gestion des risques de la **Sécurité des Systèmes d'Information** (SSI)
  - standard international lié aux Systèmes de Management de la Sécurité de l'Information (SMSI)
- ISO27k ≡ famille des standards SMSI

ISO/CEI 27000 introduction et vue globale de la famille des normes, ainsi qu'un glossaire des termes communs (mai 2009)

ISO/CEI 27001 norme de certification des SMSI (publiée en 2005)

ISO/CEI 27002 guide des bonnes pratiques en SMSI (dernière révision en 2005)
nb : précédemment connu sous le nom de ISO/CEI 17799, et avant BS 7799 Partie 1

ISO/CEI 27005 norme de gestion de risques liés à la sécurité de l'information (publiée le 4 juin 2008, révisée le 19 mai 2011) → ***Risk Manager***

ISO/CEI 27006 guide de processus de certification et d'enregistrement (publié le 13 février 2007)

Gestion des risques liés à la sécurité de l'information

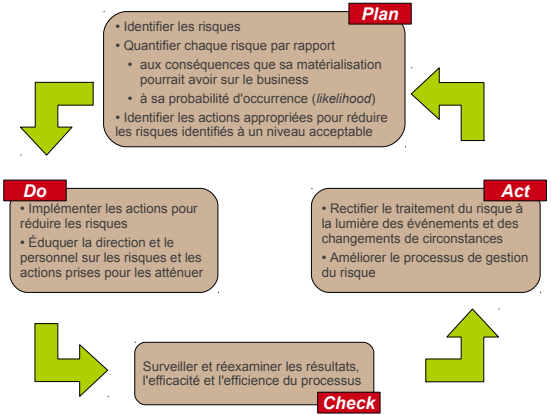
Suite ISO/CEI 27000

- Objectifs de l'ISO/CEI 27005:2011
  - la norme ISO 27005 explique en détail comment conduire l'appréciation des risques et le traitement des risques, dans le cadre de la sécurité de l'information
  - l'ISO 27005 propose une méthodologie de gestion des risques en matière d'information dans l'entreprise conforme à la norme ISO/CEI 27001
    - → *elle a donc pour but d'aider à mettre en œuvre l'ISO/CEI 27001 (certification d'un SMSI)*
  - la norme ISO 27005 peut néanmoins être utilisée de manière autonome dans différentes situations
  - elle applique à la gestion de risques le cycle d'amélioration continue PDCA

PLAN	*identification des risques, évaluation des risques et définition des actions de réduction des risques*
DO	*exécution de ces actions*
CHECK	*contrôle du résultat*
ACT	*modification du traitement des risques selon les résultats*

Gestion des risques liés à la sécurité de l'information

Suite ISO/CEI 27000



Gestion des risques liés à la sécurité de l'information

EBIOS

- **E**xpression des **B**esoins et **I**dentification des **O**bjectifs de **S**écurité
- Méthode de gestion des risques élaborée par l'ANSSI [19]
  - marque déposée par le Secrétariat général de la défense et de la sécurité nationale
  - ⇒ norme franco-française...
- Dernière màj : janvier 2010
  - → harmonisation du vocabulaire vis-à-vis des normes ISO 27001, ISO 27005,...
- Origine : rédaction de FEROS

  nb : une **f**iche d'**e**xpression **r**ationnelle des **o**bjectifs de **s**écurité est requise dans le dossier de sécurité de tout système traitant des informations classifiées

19. **A**gence **N**ationale de la **S**écurité des **S**ystèmes d'**I**nformation
http://www.ssi.gouv.fr/

Gestion des risques liés à la sécurité de l'information

EBIOS

- Objectifs
  - fournir une base commune de concepts et d'activités pragmatiques à toute personne impliquée dans la gestion des risques, notamment dans la sécurité de l'information
  - satisfaire les exigences de gestion des risques d'un système de management de la sécurité de l'information (ISO 27001)
  - définir une démarche méthodologique complète en cohérence et en conformité avec les normes internationales de gestion des risques (ISO 31000, ISO 27005,...)
  - établir une référence pour la certification de compétences relatives à la gestion des risques
- Domaine d'application
  - secteur public / secteur privé
  - petites structures (PME, collectivités territoriales,...) / grandes structures (ministère, organisation internationale, entreprise multinationale,...)
  - systèmes en cours d'élaboration / systèmes existants

Gestion des risques liés à la sécurité de l'information

EBIOS

- Positionnement par rapport aux normes
  - la méthode EBIOS respecte les exigences de l'ISO 27001 (norme d'exigences pour un SMSI)
  - elle peut exploiter les mesures de sécurité décrites dans la norme ISO 27002 (catalogue de bonnes pratiques)
  - elle est compatible avec l'ISO 31000 (cadre général pour toutes les normes sectorielles de gestion des risques)
  - c'est une méthode pour mettre en œuvre le cadre défini dans l'ISO 27005 (cadre spécifique pour gérer les risques de sécurité de l'information)
  - elle permet d'exploiter l'ISO 15408 (critères communs)

Gestion des risques liés à la sécurité de l'information

EBIOS vs. ISO 27005

- EBIOS
  - \+ gestion des risques sur le SI dans sa globalité (y compris locaux, personnes,...)
  - \- norme franco-française
  - \- contexte (trop) complet dès le départ
- ISO 27005:2011
  - \+ norme internationale
  - \+ processus de certification
  - \+ démarche incrémentale
  - \- uniquement le SI

⇒ Présentation des principes généraux de la gestion des risques dans la SSI selon la norme ISO 27005

Plan du cours

Objectif du cours

- Vous sensibiliser à une démarche de gestion des risques dans les systèmes d'information
- Sources de documentation
  - normes **ISO** (International Organization for Standardization)
    http://www.iso.org/
  - méthode **EBIOS**
    http://www.ssi.gouv.fr/
  - documents **HSC** (Hervé Schauer Consultants)
    http://www.hsc.fr/

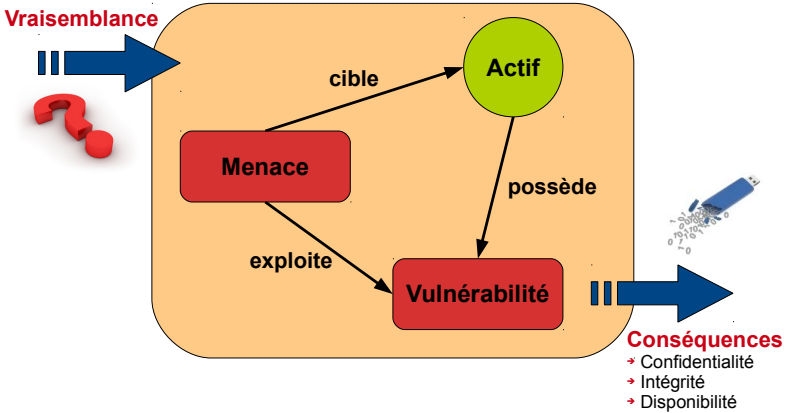
Risque

- Concept multidisciplinaire
  - assurance
    - → concept fondamental
  - droit
    - → **événement éventuel**, **incertain**, dont la réalisation ne dépend pas exclusivement de la volonté des parties et pouvant causer des dommages
  - écologie
    - → **probabilité** de survenance d'un danger
- 2 notions fondamentales
  - incertitude
  - dommage

Risque

- Risque informatique
  - → **probabilité plus ou moins grande** de voir une menace informatique se transformer en événement réel entraînant une perte
- Risque de sécurité de l'information (ISO 27005 3.2)
  - → **possibilité** (**éventualité**) qu'une menace donnée exploite une ou plusieurs vulnérabilités d'un actif ou d'un groupe d'actifs, causant ainsi des **préjudices** à l'organisme
    - il s'estime en termes de combinaison de la **vraisemblance** (probabilité d'occurrence) et de ses **conséquences**

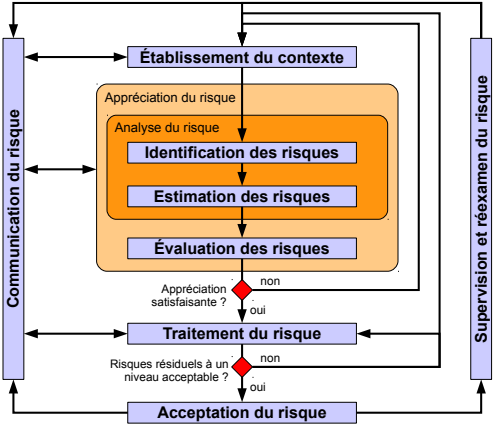
Risque en sécurité de l'information



Processus de gestion du risque

- Approche itérative
  - Établissement du contexte
  - Appréciation du risque
    - → Identification des risques
    - → Estimation des risques
    - → Évaluation des risques
  - Traitement du risque
  - Acceptation du risque
- La méthode définit aussi 2 tâches à mener en parallèle
  - Communication du risque
  - Supervision et réexamen du risque

Processus de gestion du risque



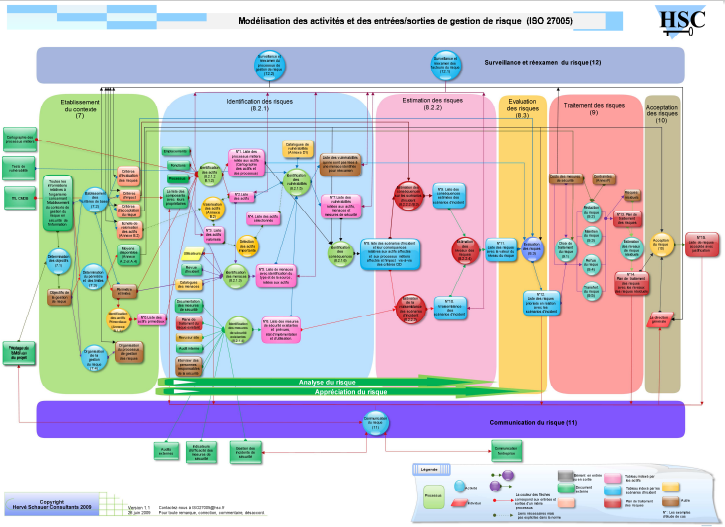
Processus de gestion du risque

- Une telle approche itérative...
  - améliore la finesse de l'analyse à chaque itération
  - fournit une bonne répartition entre le temps et l'effort fourni pour identifier les mesures de sécurité
  - permet de traiter les risques en fonction des ressources et des moyens qui sont disponibles
  - facilite les liens entre les risques et les conséquences sur les processus métier
  - permet d'avancer lorsque les interlocuteurs sont absents ou les livrables incomplets
  - facilite la gestion des susceptibilités et des aspects politiques (...) entre les interviewés, les propriétaires d'actifs et de processus métier
  - tend progressivement vers une maîtrise des risques de haut niveau et qui soit conforme aux besoins de l'organisme

Présentation de la méthode

- Une bonne présentation des principes essentiels de la méthode ISO 27005
  → **Méthode de management des risques ISO 27005**
  Hervé Schauer Consultants
  Paris, 15 avril 2010

  - schéma modélisant chaque activité et sous-activité de la méthode proposée par la norme ISO 27005
  - cas d'étude illustrant les différentes étapes de la méthode
  - toutes les explications indiquent les références aux sections de la norme


Modélisation des activités et des entrées/sorties de gestion de risque (ISO 27005)
HSC
Surveillance et réexamen du risque (12)
Etablissement du contexte (7)
Identification des risques (8.2.1)
Estimation des risques (8.2.2)
Evaluation des risques (8.3)
Traitement des risques (9)
Acceptation des risques (10)
Analyse du risque
Appréciation du risque
Communication du risque (11)
Copyright
Hervé Schauer Consultants 2009
Version 1.1
28 juin 2009
Contactez-nous à SECURITE@hsc.fr
Pour toute remarque, correction, commentaire, désaccord.
Légende
Processus
Activité
Document
Information
Tableau indiquant les actifs
Tableau indiquant les menaces
Tableau indiquant les vulnérabilités
Mail du traitement des risques
Pulse
N° Lots exemples définis ds cas

HSC Formations

here

Plan du cours

Droit & Numérique

- La prise en compte des aspects juridiques en SSI est multiple :
  - Droit d'auteur & propriété intellectuelle
  - Liberté d'expression (et ses limites)
  - Contrat de travail & obligation de loyauté envers l'employeur
  - Respect des différentes réglementations en vigueur
  - **Protection des données personnelles**

Les textes fondateurs

- Loi n° 78-17 du 6 Janvier 1978 relative à l'informatique, aux fichiers et aux libertés
- Directive européenne n° 95/46/CE du 24 octobre 1995
- Charte des droits fondamentaux de l'Union Européenne
  - → Chapitre II : libertés
- Règlement Général pour la Protection des Données (RGPD)
  - → entré en vigueur le 25 mai 2018
  - → règlement européen
    - ⇒ pas de transposition nationale
    - ⇒ est "obligatoire dans tous ses éléments et directement applicable dans tout État membre"

CNIL

- La France, pionnière en matière de données personnelles, a très vite réalisé les risques de l'informatique pour les libertés de la personne humaine et a réagi en se dotant depuis 1978 d'une législation traitant de la protection de ces données face aux dangers d'une informatique grandissante : la loi du 6 janvier 1978 dite "informatique et libertés".
- La Commission Nationale de l'Informatique et des Libertés (CNIL) est une autorité administrative indépendante née de l'adoption de la loi précitée.
- La mission générale de la CNIL est, face aux dangers de l'informatique, de protéger la vie privée et les libertés individuelles ou publiques.

CNIL

- La CNIL est une autorité administrative indépendante qui fonctionne avec une dotation du budget de l'État.
- Elle informe et conseille les personnes sur leurs droits : elle reçoit les réclamations, pétitions, plaintes relatives aux traitements de données personnelles et répond par des avis, délibérations ou recommandations.
- Elle régule et recense les fichiers de données personnelles : elle autorise ou non la création des fichiers.
- Elle peut opérer des contrôles dans les entreprises, des enquêtes, des auditions de personnes.
- La CNIL n'est pas un juge. Il faut saisir le juge pour obtenir des dommages-intérêts au titre de la responsabilité civile en cas de préjudice.

Notion de données sensibles

- **Attention** (→ responsables de traitements) : En pratique une confusion est parfois opérée entre :
  - *"les données qui sont sensibles pour une organisation"* (ex : des données financières pour une entreprise)
  - le régime juridique des *"données sensibles, au sens de la loi de 1978"*
- Des données sensibles pour une entreprise (l'évolution de son chiffre d'affaire, ses processus d'acquisition de clientèle,...) ou une administration (organisation interne, dossiers politiquement et médiatiquement sensibles,...) ne sont pas nécessairement des données sensibles au sens de la loi de 1978.

Notion de données sensibles

- Cette distinction est importante car seul le traitement de *données sensibles au sens de la loi informatique et libertés* devra répondre au régime juridique décrit ci-après.
- Peu importe qu'une organisation traite des *données qui lui sont sensibles*, dès lors que ces données ne sont pas des données personnelles.

Droits et principes fondamentaux

Loi "informatique et libertés"

La loi de 1978, dite loi "informatique et libertés", instaure de nombreux droits et principes fondamentaux relatifs **"à la protection des personnes physiques à l'égard des traitements de données à caractère personnel"**.

- Principe de **finalité**
  - Les données personnelles ne peuvent être collectées, traitées, conservées ou transmises à des tiers qu'en vue de réaliser des finalités déterminées, légitimes et compatibles entre elles.

Droits et principes fondamentaux

- Principe de **loyauté** et de **transparence**
  - La collecte, le traitement, la conservation des données personnelles et leur transmission éventuelle à des tiers doivent s'effectuer de manière loyale.
  - Cela suppose que les données ne soient pas collectées et traitées à l'insu de la personne concernée et que les personnes soient informées de l'identité et du lieu d'établissement de la personne qui traite ces données, des finalités poursuivies, du caractère obligatoire ou facultatif du traitement des données, des destinataires des informations, ainsi que toute information nécessaire à l'exercice de leurs droits.

Droits et principes fondamentaux

- Principe de la **pertinence** et de l'**exactitude** des données
  - Les données personnelles faisant l'objet d'un traitement doivent être pertinentes au regard des finalités poursuivies. Elles doivent être exactes et mises à jour.
- Principe du **consentement** pour les traitements de données sensibles
  - Lorsque des traitements portent sur des données sensibles (religion, opinion politique ou philosophique, appartenance syndicale, origine raciale et ethnique, santé et vie sexuelle), celles-ci ne peuvent être collectées qu'avec le consentement des personnes.

Droits et principes fondamentaux

- Principe d'**accès**, de **rectification** et d'**opposition**
  - Les personnes doivent se voir reconnaître les droits d'accéder, sans subir de coût dissuasif, à toute donnée les concernant, de corriger les données incomplètes ou inexactes et de s'opposer sans avoir à se justifier à l'exploitation de leurs données à des fins commerciales.
- Principe de **sécurité**
  - Le code pénal incrimine le traitement, sans que soient prises les mesures de précaution, et prévoit des sanctions à l'encontre de l'administrateur ne protégeant pas assez efficacement son système (délit de manquement à la sécurité).

Droits et principes fondamentaux

- Principe du **droit à l'oubli**
  - Le code pénal incrimine le fait, sans l'accord de la CNIL, de conserver une information sous la forme nominative au-delà de la durée prévue à la demande d'avis ou à la déclaration préalable.
- Principe de **protection de la considération et de l'intimité**
  - Le fait de porter à la connaissance d'un tiers des images portant atteinte à la considération de l'intéressé ou à l'intimité de sa vie privée est condamnable.
  - Cette divulgation est sanctionnée encore plus sévèrement si elle a été faite par imprudence ou négligence (délit d'atteinte à la considération ou à l'intimité).

Résumé droits & obligations

- Les droits des personnes
  - **Droit à l'information**
    Toute personne a le droit de savoir si elle est fichée et dans quels fichiers elle est recensée.
  - **Droit d'accès**
    [...] a le droit d'interroger le responsable d'un fichier pour savoir s'il détient des informations sur elle et d'en obtenir la communication.
  - **Droit de rectification**
    [...] a le droit de contrôler l'exactitude des données et de les faire rectifier.
  - **Droit d'opposition**
    [...] peut s'opposer pour des motifs légitimes à figurer dans un fichier ou de voir communiquer des informations sur elle à des tiers. Les personnes peuvent saisir la CNIL en cas de difficultés dans l'exercice de leurs droits.

Résumé droits & obligations

- Les obligations des responsables du traitement
  - **Obligation d'information préalable** des personnes concernées dont on doit obtenir le consentement exprès.
  - **Obligation d'assurer la sécurité et la confidentialité** des données collectées et traitées.
  - **Obligation d'une collecte et d'un traitement** ayant une finalité précise et effectués de façon licite et loyale.
  - **Obligation de déclaration préalable à la CNIL** des traitements informatiques de données personnelles.

Données à caractère personnel

- Points clés
  - Les données sont des données à caractère personnel dès lors qu'elles portent sur une **personne identifiée ou identifiable**, la personne concernée.
  - Une personne est identifiable si des informations complémentaires peuvent être obtenues sans effort déraisonné, permettant l'identification de la personne concernée.
  - L'authentification s'entend du fait de démontrer qu'une certaine personne possède une certaine identité et/ou est autorisée à exercer certaines activités.
  - NB : **identification $\neq$ authentification**

Données à caractère personnel

- Points clés
  - Il existe des catégories particulières de données, appelées "données sensibles", énumérées dans la directive relative à la protection des données, qui requièrent une protection accrue et, par conséquent, sont soumises à un régime juridique spécial.
  - Les données sont anonymisées si elles ne contiennent plus d'identifiants ; elles sont pseudonymisées si les identifiants sont cryptés.
  - Contrairement aux données anonymisées, les données pseudonymisées sont des données à caractère personnel.

Caractère identifiable d'une personne

- Dans le droit de l'UE, une information contient des données sur une personne si :
  - ▷ une personne est identifiée dans cette information

  ou ▷ si une personne, bien que non identifiée, est décrite dans cette information d'une manière permettant de découvrir qui est la personne concernée en menant d'autres recherches

- Les deux types d'informations sont protégés de la même manière par le droit européen en matière de protection des données.
  - noms non uniques ⇒ date et lieu de naissance, numéros de citoyens,...
  - ère du numérique ⇒ données biométriques (empreintes digitales, photos numériques, aspects rétiniens) pour l'identification des personnes

Authentification

- L'authentification est la procédure par laquelle une personne peut prouver qu'elle possède une certaine identité et/ou est autorisée à faire certaines choses.
  - Par la comparaison de données biométriques (une photo ou les empreintes digitales d'un passeport) avec les données de la personne qui se présente à un contrôle d'immigration.
  - En demandant des informations que seule la personne possédant une certaine identité ou autorisation devrait connaître (numéro d'identification personnel (PIN) ou un mot de passe).
  - En demandant la présentation d'un certain objet qui devrait exclusivement se trouver en la possession de la personne ayant une certaine identité ou autorisation (une carte magnétique spéciale ou la clé d'un coffre en banque).

Authentification

- L'authentification est la procédure par laquelle une personne peut prouver qu'elle possède une certaine identité et/ou est autorisée à faire certaines choses.
  - Outre les mots de passe ou cartes magnétiques, parfois associés à des codes PIN, les signatures électroniques sont un outil particulièrement utile pour identifier ou authentifier une personne dans des communications électroniques.
- NB : L'authentification ne nécessite pas de stocker les données personnelles (ex : empreinte digitale) sur le serveur, contrairement à l'identification.

Catégories particulières de données à caractère personnel

- Il existe des catégories particulières de données qui, par leur nature, peuvent faire courir un risque aux personnes concernées quand elles font l'objet d'un traitement et requièrent donc une protection accrue :
  - données à caractère personnel révélant l'origine raciale ou ethnique
  - données à caractère personnel révélant les opinions politiques, convictions religieuses ou autres convictions
  - données à caractère personnel relatives à la santé ou à la vie sexuelle

Données anonymisées et pseudonymisées

- Principe de la conservation des données pendant une durée limitée :
  - ▷ les données doivent être conservées *"sous une forme permettant l'identification des personnes concernées pendant une durée n'excédant pas celle nécessaire à la réalisation des finalités pour lesquelles elles sont collectées ou pour lesquelles elles sont traitées ultérieurement"*

⇒ Il pourrait être nécessaire d'anonymiser des données si un responsable du traitement souhaite les conserver alors qu'elles ne sont plus d'actualité et qu'elles ne servent plus leur finalité initiale.

Données anonymisées et pseudonymisées

- **Données anonymisées**
  - Des données sont anonymisées si tous les éléments identifiants ont été supprimés d'un ensemble de données à caractère personnel.
  - Les informations ne doivent plus contenir aucun élément qui soit susceptible, au moyen d'un effort raisonnable, de servir à réidentifier la ou les personnes concernées.
  - Lorsque des données ont été correctement anonymisées, elles ne sont plus des données à caractère personnel.

Données anonymisées et pseudonymisées

- **Données pseudonymisées**
  - Les informations personnelles contiennent des identifiants, tels que le nom, la date de naissance, le sexe ou l'adresse.
  - Lorsque des informations personnelles sont pseudonymisées, les identifiants sont remplacés par un pseudonyme.
  - La pseudonymisation est notamment obtenue par cryptage des identifiants figurant dans les données à caractère personnel.
  - Il ne doit pas être possible de relier facilement les données et les identifiants. Pour quiconque ne possède pas la clé de décryptage, les données pseudonymisées peuvent être difficilement identifiables.
  - Le lien avec l'identité demeure sous la forme du pseudonyme associé à la clé de décryptage. Pour toute personne habilitée à utiliser la clé de décryptage, une nouvelle identification est possible aisément ⇒ **données personnelles**

Une constante évolution

- Règlement Général pour la Protection des Données (**RGPD**) le 25 mai 2018
- Règlement **ePrivacy** (25 mai 2018 ?)
  - → réforme la directive 2002/58/CE du 12 juillet 2002
  - → relatif au traitement des données à caractère personnel et la protection de la vie privée dans le secteur des communications électroniques
- Plusieurs référentiels de sécurité informatique seront rendus opposables "dès 2018"
  - → source Agence des Systèmes d'Information Partagés de santé (Asip santé)
  - → ex : référentiels d'interopérabilité

RGPD

- Rien de révolutionnaire !
  - en France nous avions déjà la loi "informatique et libertés"
  - modifiée par la loi du 6 août 2004 afin de transposer en droit français les dispositions de la directive 95/46/CE
- Dans les grandes lignes, le RGPD cherche à renforcer la responsabilité des sociétés amenées à gérer des informations personnelles. Ses différentes dispositions cherchent donc à assurer la protection de ces données, mais aussi leur traçabilité, et le suivi précis des traitements qui en seront faits.

RGPD

- Le poste de **DPO** (Data Protection Officer) s'inscrit dans le prolongement de ce que la CNIL avait déjà initié avec son Correspondant Informatique et Libertés (**CIL**), avec un niveau de responsabilité similaire mais des prérogatives étendues.
- Ce pilote en interne est censé cartographier l'ensemble des traitements de données personnelles réalisés par l'entreprise pour identifier les carences et proposer une optimisation des processus.

NB : La CNIL recommande également la création d'une documentation permettant de justifier des mesures entreprises en cas d'enquête de conformité.

RGPD & PIA

- L'Étude d'Impact sur la Vie Privée (**EIVP**) était la traduction utilisée par la CNIL pour le Privacy Impact Assessment (**PIA**).
- L'application du nouveau règlement européen RGPD impose (en partie) la réalisation d'une analyse d'impact relative à la protection des données (ou **DPIA** pour Data Protection Impact Assessment).
- Cette démarche, repose sur une analyse de risques sécurité orientée uniquement sur les risques visant les données personnelles et leurs impacts sur les droits et libertés des personnes concernées par ces données.

RGPD & PIA

- Le DPIA n'est pas obligatoire pour l'ensemble des traitements, mais uniquement pour les traitements présentant *"un risque élevé pour les droits et libertés des personnes physiques"*
  - → les traitements à grande échelle
  - → la surveillance systématique à grande échelle d'une zone accessible au public (notamment la vidéosurveillance)
  - → les décisions automatiques produisant des effets juridiques (pour des offres de prestations, ou le choix de contractualisation)
  - → le traitement de données sensibles (données de santé, opinions politiques, orientation sexuelle)
  - → l'évaluation ou la notation basée sur des données personnelles, y compris le profilage et la prédiction
  - → le traitement de données biométriques, de données relatives à des condamnations pénales et à des infractions

ePrivacy

- Là où le RGPD s'attache à la protection des données personnelles dans leur ensemble, ePrivacy se concentrera plus précisément sur l'exploitation des données issues des communications.
  - Avec ce texte, la Commission européenne affiche sa volonté de remettre au premier plan la notion de consentement de l'internaute, en particulier pour ce qui concerne la question des cookies.
  - Il sera difficile d'esquiver son impact sur les pratiques en matière de marketing...

ePrivacy

- Aujourd'hui, la norme est au cas par cas, ce qui signifie que chaque éditeur se charge de recueillir le consentement du visiteur avant de distribuer les traceurs dédiés aux opérations de ciblage.
- Demain, le règlement ePrivacy prévoit que le consentement se fasse au niveau des options du navigateur Web, de façon globale.
- Lourde de conséquence pour l'ensemble des acteurs du Web, la mesure sera âprement discutée jusqu'à l'adoption définitive du règlement ePrivacy, prévue pour la fin de l'année 2017. Il ne restera ensuite que six mois pour se mettre en conformité...

Domaine de la santé

- Le 6 octobre 2017, le directeur général de l'Agence des systèmes d'information partagés de santé (Asip santé) a assuré qu'*"au moins trois référentiels de sécurité informatique seront publiés par arrêtés ministériels et rendus opposables dès 2018"*.
- Élaboration d'un cadre d'interopérabilité des systèmes d'information de santé, la création d'un espace de confiance "pour l'ensemble des interactions supposées par l'e-santé", notamment via :
  - les messageries sécurisées de santé (MSSanté)
  - l'adoption d'une politique générale de sécurité des systèmes d'information de santé (PGSSI-S)

Domaine de la santé

- Parmi les référentiels qui seront rendus opposables dès 2018 par arrêté ministériel (dixit) :
  - l'un porte sur l'identification
  - l'autre sur l'authentification
  - le troisième sur la gouvernance de la sécurité
- Parallèle avec les actions menées sur la sécurité informatique
  - → la loi prévoit aussi que les référentiels d'interopérabilité soient rendus opposables juridiquement

Partage de données

- Avec l'émergence des environnements connectés, IoT, smart-*, IA & ML le législateur propose de nouveaux règlements :
  - **RGPD** → Règlement Général sur la Protection des Données (27/04/2016 ⇝ 25/05/2018)

    https://fr.wikipedia.org/wiki/Règlement_général_sur_la_protection_des_données
  - **LRN** → Loi pour une République Numérique (07/10/2016)

    https://fr.wikipedia.org/wiki/Loi_pour_une_République_numérique
  - **DGA** → Data Governance Act (25/11/2020)

    https://eur-lex.europa.eu/legal-content/FR/ALL/?uri=CELEX%3A52020PC0767
  - **Une stratégie européenne pour les données** (19/02/2020)

    https://eur-lex.europa.eu/legal-content/FR/TXT/?uri=CELEX%3A52020DC0066

Partage de données

- Voire également de nouveaux concepts :
  - **Open Data**
  - **Communs numériques**

Plan du cours

D'autres facettes de la sécurité informatique

- Dans ce cours nous nous sommes essentiellement intéressés à la sécurité informatique au niveau applicatif
  - → règlement & politique de sécurité
  - → sécurité dans les bases de données
  - → sécurité dans les applications Java
- Voire même à un niveau encore supérieur...
  - → gestion des risques dans les systèmes d'informations
  - → droit & numérique
- Mais la Sécurité des Systèmes d'Information (SSI) recours à bien d'autres techniques & technologies

Sécurité réseau

- Modèle OSI (Open Systems Interconnect) en 7 couches

application c'est le programme qui a besoin du réseau pour communiquer
ex : navigateur web (HTTP), logiciel de messagerie (POP, IMAP, SMTP) ou de transfert de fichier (FTP), telnet,...

présentation responsable de la représentation des données (de telle sorte qu'elle soit indépendante du type de microprocesseur ou du système d'exploitation par exemple) et, éventuellement, du chiffrement (ex : HTML)

session en charge d'établir et maintenir des sessions (ie. débuter le dialogue entre 2 machines : vérifier que l'autre machine est prête à communiquer, s'identifier,...)

transport en charge de la liaison d'un bout à l'autre ; s'occupe de la fragmentation des données (fichiers) en petits paquets et vérifie éventuellement qu'elles ont été transmises correctement et dans l'ordre (ex : TCP, UDP)

réseau en charge du transport, de l'adressage et du routage des paquets entre extrémités distantes, à travers le réseau (ex : IP)

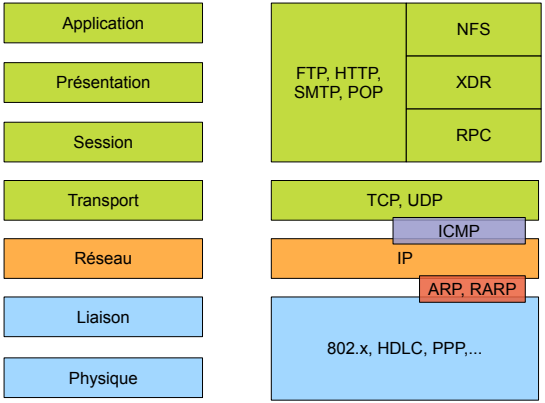
liaison en charge de transmettre des trames ; fournit également la détection d'erreur (retransmission) et la synchronisation (ex : transmission de trames entre nœuds d'un segment Ethernet)

physique responsable de la transmission de bits : codage, modulation,... (ex : normes des modems V23, V92,...)

support c'est le support de transmission lui-même : un fil de cuivre, une fibre optique, les ondes hertziennes, une liaison infrarouge,...

Sécurité réseau

- Comparaison modèle OSI & modèle TCP/IP



Sécurité réseau

- De nombreux mécanismes existent pour sécuriser le réseau du système d'information
  - sécurité du médium de communication
    - → *Wi-Fi sécurisé (ex : WPA & WPA2), sauts de fréquence,...*
  - architecture du réseau
    - → *VLANs, routage, firewall, DMZ, NAT, PAT, VPN,...*
  - sécurité des services réseau
    - → *SSL, HTTP, FTP, WebDAV, NFS, Samba, DNS, DHCP, SSH, telnet, NTP, proxy,...*
  - supervision du réseau
    - → *SNMP, Nagios, Syslog, IDS,...*

Sécurité système

- Bien évidemment, la sécurité informatique concerne également l'administration système
  - système d'exploitation
    - → *mises à jour de l'OS, patchs de sécurité, services activés,...*
  - logiciels liés à la sécurité
    - → *antivirus, détection des fichiers système modifiés, surveillance des processus exécutés,...*
  - politique de sécurité
    - → *gestion des utilisateurs & des groupes, permissions, GPO sous Windows,...*
    - → *horaires & lieux de connexion, traçabilité,...*

Sécurité système

- Sécurité & aspects matériels
  - tolérance aux pannes
    - → *onduleur & alim. de secours, alim. redondante, RAID*
    - → *virtualisation, clusters de serveurs, multi-sites,...*
  - reprise après "sinistre" la + rapide possible
    - → *gestion des sauvegardes*
    - → *maquettes pour réinstaller les OS, les applis, les licences*
  - sécurité des locaux
    - → *accès, protection incendie, inondation,...*
  - renouvellement du matériel
    - → *garantie, suivi constructeur, dispo. des mises à jour,...*
    - → *anticiper les besoins, prévoir les évolutions,...*

Sécurité applicative

- Certains mécanismes de sécurité son implémentés au niveau des couches applicatives
  - protocoles de communication
    - → *RMI*, *CORBA*, *Web Services*, *WSS*,...
      - ▷ chiffrement des échanges
      - ▷ signatures & certificats ⇒ authentification de l'émetteur & intégrité
      - ▷ transaction, session, *security token*,...
      - ▷ "intercepteurs" pour mettre en place des politiques de sécurité particulières
  - mécanismes d'authentification applicative
    - → *SSO*, *CAS*, *Radius*,...
      - ▷ authentification unique de l'utilisateur ⇒ session
      - ▷ *security token* ⇒ attributs, permissions,...
      - ▷ objectif : décharger les applis web & services réseau de l'authentification

Communication & formation

- Comme le précisent clairement les différentes méthodes de gestion des risques (quel que soit le domaine d'ailleurs), la sécurité ne peut être assurée sans la "participation" des utilisateurs
  - communication
    - → *expliquer/justifier les mécanismes mis en œuvre*
    - → *informer les utilisateurs des enjeux, des ressources sensibles*
    - → *établir des procédures*
    - → *rassurer & mettre en confiance les partenaires, clients,...*
  - formation
    - → *former les utilisateurs aux outils & logiciels*
      - ▷ limiter les erreurs liées à une mauvaise utilisation
    - → *sensibiliser les utilisateurs aux objectifs de sécurité (cf. CID) pour certaines données*
      - ▷ appliquer la PSSI, les procédures,...

Merci de votre attention